Вінницький національний технічний університет
(повне найменування вищого навчального закладу)
Факультет інтелектуальних інформаційних технологій та автоматизації
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна технологія контролю прийняття ліків»

Виконав: студент 2-го курсу, групи 1КН-22м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
_____ Русначенко М. О.
(прізвище та ініціали)

Керівник: к. т. н., доцент каф. КН
_____ Арсенюк І. Р.
(прізвище та ініціали)
«07» 12 2023

Опонент: к. т. н., доцент каф. САІТ
_____ Жуков С. О.
(прізвище та ініціали)
«07» 12 2023

Допущено до захисту
Завідувач кафедри КН
д. т. н., проф. Яровий А. А.
(прізвище та ініціали)
«08» 12 2023 р.

Вінниця ВНТУ - 2023 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 122 «Комп'ютерні науки»
Освітньо-професійна програма – «Системи штучного інтелекту»

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
Д. т. н., проф. Яровий А. А.
29.08 2023 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Русначенку Михайлу Олександровичу

(прізвище, ім'я, по батькові)

1. Тема роботи: Інформаційна технологія контролю прийняття ліків.

керівник роботи: к. т. н., доцент кафедри КН Арсенюк І. Р.
затверджені наказом вищого навчального закладу від «18» вересня 2023 року № 247
2. Строк подання студентом роботи «13» листопада 2023 року
3. Вихідні дані до роботи:
Потужність бази даних лікарських препаратів – не менше ніж 300000 записів, кількість показників стану здоров'я користувачів – не менше 3 одиниць, тестова вибарка – не менше 100 користувачів, мова програмування – функціональна, тип застосунку – кросплатформний
4. Зміст текстової частини:
Вступ, аналіз предметної області та обґрунтування доцільності розробки інформаційної технології контролю прийняття ліків, проєктування інформаційної технології контролю прийняття ліків, програмна реалізація інформаційної технології контролю прийняття ліків, економічна частина, висновки, перелік використаних джерел, додатки.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень)
Алгоритм роботи програми контролю прийняття ліків, UML діаграма класів програми контролю прийняття ліків, робочі вікна програми контролю прийняття ліків, результати тестування програми контролю прийняття ліків.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціалита посада консультанта	Підпис, дата	
		завдання видав	викон прий
1-3	Арсенюк І. Р., к. т. н., доц. каф. КН	29.09.23	10.11.23
4	Адлер О. О., к. т. н., доц. каф. ЕПВМ	16.10.23	5.11.23

7. Дата видачі завдання 29.08 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Пр
1	Аналіз сучасного рівня інформаційних технологій контролю прийняття ліків. Постанова задач дослідження	01.09.23 – 07.09.23	
2	Розробка алгоритму інформаційної технології	08.09.23 - 24.09.23	
3	Практичне застосування та оцінака ефективності розробленого алгоритму	25.09.23 – 15.10.23	
4	Підготовка економічної частини	16.10.23 - 05.11.23	
5	Апробація результатів дослідження	21.10.23 - 05.11.23	
6	Оформлення пояснювальної записки, графічного матеріалу, презентації	06.11.23 – 10.11.23	

Студент _____ (підпис) Русначенко М. О.

Керівник роботи _____ (підпис) Арсенюк І. Р.

АНОТАЦІЯ

УДК 004.8

Русначенко М. О. Інформаційна технологія контролю прийняття ліків. Магістерська кваліфікаційна робота зі спеціальності 122 – комп'ютерні науки, освітня програма – системи штучного інтелекту. Вінниця: ВНТУ, 2023. 97 с.

На укр. мові. Бібліогр.: 30 назв; рис.: 20; табл. 8

Дана магістерська кваліфікаційна робота присвячена розробці програмного забезпечення для контролю прийняття ліків. Проаналізовано комерційні рішення контролю прийняття ліків, запропоновано новий формат програмного продукту для виконання даної задачі.

Модефіковано існуючу математичну модель та розроблено алгоритм для системи контролю прийняття ліків, обрано експертну систему та інтегровано у систему. Обґрунтовано мову програмування та середовище розробки даної системи. Створено мобільний застосунок для контролю прийняття ліків на мові JavaScript.

Графічна частина складається з 6 плакатів.

У економічному розділі розраховано суму витрат на розробку та виготовлення нового технічного рішення, яка складає 160184 гривень, спрогнозовано орієнтовану величину витрат по кожній з статей витрат, розраховано чистий прибуток, термін окупності витрат для виробника 0,74 роки та економічний ефект для споживача при використанні даної розробки.

Ключові слова: контроль лікування, прийняття ліків, експертна система, мікросервіс.

ABSTRACT

Rusnachenko M. O. Information technology for control of medication administration. Master's thesis in the specialty 122 - computer science, educational program – artificial intelligence systems. Vinnytsia: VNTU, 2022. 97 p.

In Ukrainian language. Bibliographer: 30 titles; fig.: 20; table 8.

This master's qualification work is devoted to the development of software for monitoring the acceptance of medications. Commercial solutions for drug administration control were analyzed, and a new software product format for this task was proposed.

The existing mathematical model was modified and an algorithm was developed for the drug administration control system, an expert system was selected and integrated into the system. The programming language and development environment of this system are substantiated. A mobile application for medication administration control was created in JavaScript. The graphic part consists of 6 posters.

The economic section calculates the amount of costs for the development and manufacture of a new technical solution, which is 160184 hryvnia, predicted the estimated cost of each of the cost items, calculated net profit, payback period for the manufacturer 0.74 years and the economic effect for consumers using this development.

Keywords: control, medicine, machine learning, decision tree.

ЗМІСТ

ВСТУП

Актуальність. Контролювання прийняття ліків, може бути корисним для людей, які п’ють ліки щодня. Це може запобігти серйозні негативні наслідки для здоров’я.

Прийом ліків у потрібний час покращує їх ефективність та безпечність.

Одне дуже важливе дослідження 2022 року показало, що учасники відкладають прийом ліків у 80 – 85% випадків і забувають прийняти їх у 44 – 46% випадків. Також велика кількість пацієнтів не звертається до лікаря у випадку погіршення стану під час лікування. Неприймання ліків за призначенням призводить до приблизно 125000 смертей щороку [2].

В наш час зростає кількість хронічних захворювань таких як діабет, серцеві захворювання та гіпертонія, які вимагають тривале лікування та жорсткий контроль у прийнятті ліків. У цих випадках є ефективним контроль ліків за допомогою технологій. Тим паче зараз широко використовуються смартфони і високий рівень їхньої доступності. Багато пацієнтів уже володіють пристроями, які можуть запускати мобільні застосунки.

Існує багато порад і рекомендацій, які допоможуть контролювати прийом ліків, але найбільш дієвими є програми, які можна завантажити, які надають щоденні нагадування за допомогою смартфона, планшета чи розумного годинника. Такі програми допомагають автоматизувати та відстежувати дози, тому ви з меншою ймовірністю можете пропустити прийом медичних препаратів.

Зв'язок роботи з науковими програмами, планами, темами. Магістерська кваліфікаційна робота виконана відповідно до напрямку наукових досліджень кафедри комп’ютерних наук Вінницького національного технічного університету 23 К1 «Моделі, методи, технології та пристрої інтелектуальних інформаційних систем управління, економіки, навчання та комунікацій» та плану наукової та навчально-методичної роботи кафедри.

Мета та завдання дослідження. Метою магістерської кваліфікаційної роботи є розширення функціональних можливостей програмного додатку.

Для досягнення поставленої мети слід розв’язати такі завдання:

- проаналізувати предметну область, сучасний стан розвитку програмних продуктів для контролю прийняття ліків;
- розглянути існуючі методи контролю прийняття ліків та обґрунтувати вибір методу, що задовільнять мету даної магістерської роботи;
- розробити експертну систему для контролю прийняття ліків;
- розробити структуру, алгоритми роботи програмного забезпечення;
- розробити відповідне програмне забезпечення;
- виконати тестування програмного забезпечення та проаналізувати отримані результати.

Об’єкт дослідження – процес контролю прийняття ліків.

Предмет дослідження – інформаційна технологія та програмні засоби для контролю прийняття ліків.

Методи дослідження. У роботі використані методи та підходи до розробки інтелектуальних систем, методи та підходи до розробки мобільних застосунків, методи функціонального програмування.

Наукова новизна одержаних результатів полягає в наступному:

Набула подальшого розвитку інформаційна технологія контролю прийняття ліків, в основі якої покладена математична модель композитного показинка добробуту, що дозволило підвищити точність роботи програмних засобів контролю прийняття ліків.

Практичне значення одержаних результатів полягає у тому, що на основі проведених досліджень розроблено програмне забезпечення контролю прийняття ліків. Запропоноване програмне забезпечення сприяє покращенню точності прийняття ліків.

Розроблено алгоритм роботи програмного забезпечення та інформаційну технологію контролю прийняття ліків.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується коректністю постановки завдання, коректністю використання математичного апарату методів дослідження, експериментальними дослідженнями тестування програмної реалізації інформаційної технології контролю прийняття ліків. Адекватність розроблених математичних моделей підтверджується результатами експериментальних досліджень.

Особистий внесок магістранта. Усі результати, наведені у магістерській кваліфікаційній роботі, отримані самостійно. У працях, які написано у співавторстві, здобувачу належать: перспективність використання мікрофронтендів для реалізації застосунку контролю прийняття ліків [1].

Апробація результатів роботи. Результати роботи були апробовані на конференції «Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення (випуск 82)», м. Тернопіль, Україна, м. Ополе, Польща, 9 - 10 листопада 2023 року [1].

Публікації. За результатами дослідження опубліковано тези доповіді на науково-технічній конференції [1] та подано заяву на авторське право на програмне забезпечення.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОБҐРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ КОНТРОЛЮ ПРИЙНЯТТЯ ЛІКІВ

1.1 Аналіз предметної області

Інформаційні технології зробили величезний внесок у сучасне представлення нашого суспільства. Важко знайти сферу, яка б наразі не використовувала їх. Широке використання комп’ютерних технологій має місце в галузі архітектури, освіти і, звичайно, медицини.

Інформаційні технології все ширше використовуються в охороні здоров’я, що іноді навіть необхідно. Це вивело медицину на новий рівень і надає їй особливих властивостей.

Багато медичних досліджень у всьому світі сьогодні неможливі без комп’ютера та спеціального програмного забезпечення. Воно супроводжується глобальними змінами в теорії та практиці медицини, які пов'язані з внесенням змін на етапі підготовки лікарів і безпосередньо в медичну практику.

Протягом нашого життя, так чи інакше, люди взаємодіють з різними лікарями, довіряючи їм наше здоров’я і життя. Але сьогодення частково змінює способи звичного взаємодії суспільства. Багато в чому це пов'язано з розвитком інформаційних технологій.

І навіть коли наявність IT для пацієнта стала помітною, це лише мала частина всього цього. Тож головне питання – що спільного між медициною та комп’ютерними технологіями і як вони сьогодні пов’язані?

Наприклад, у Сполучених Штатах UnitedHealth Group і Cisco утворюють національну телемедичну мережу, яка з’єднує технології багатьох постачальників, системи електронної історії хвороби та інші інформаційні платформи. Це має забезпечити спілкування та консультації в реальному часі між лікарями, медсестрами та іншими спеціалістами по всій країні. З його

допомогою пацієнти зможуть скористатися перевагами спеціалізованих та профілактичних послуг, таких як лікування широкого спектру від застуди, грипу, ГРЗ та алергії до хронічних захворювань (цукровий діабет, гіпертонія та хвороби серця, які потребують допомоги спеціаліста); повторна діагностика пацієнтів, які мали специфічні захворювання або травми; профілактичні фітнес-програми для персоналу та пацієнтів; надання невідкладної медичної допомоги. Цю тенденцію вже впроваджують інші країни світу.

Україна наполегливо і дуже активно залучає ІТ у різні сфери. Значним досягненням стало створення eHealth – електронної системи охорони здоров'я, що забезпечує обмін медичною інформацією та реалізацію програми медичних гарантій для населення. Ця система допомагає пацієнтам отримувати якісні медичні послуги та дає можливість їх надавати лікарям. Крім того, він має можливість контролювати ефективність державних витрат на охорону здоров'я, таким чином запобігаючи зловживанням [3].

По-перше, система має включати відділення первинної допомоги, до складу якого входять сімейні лікарі, терапевти та педіатри. Пацієнти зможуть подавати декларації обраним лікарям, а лікарі реєструватимуть їх у цій системі. В результаті держава оплачує лікарям за кожного пацієнта, а пацієнт гарантовано отримує медичні пільги, які держава надаватиме безкоштовно. Таким чином, електронне здоров'я дотримується принципу, який уже широко використовується в усьому світі, а саме «гроші йдуть за пацієнтом». У перспективі розвиток системи eHealth: забезпечення того, щоб кожен міг швидко отримати медичну інформацію, а для лікарів – правильний діагноз з урахуванням цілісної картини стану здоров'я своїх пацієнтів. Лікарі призначають ліки в електронному вигляді. Система буде містити всю історію хвороби пацієнтів і буде доступна їм та їхнім лікарям [4].

Створення цієї системи є найважливішим кроком у досягненні «єдиного медичного простору», результатом якого стане абсолютна координація та інтеграція між різними рівнями охорони здоров'я. Це також ефективний внесок у впровадження системи управління якістю послуг.

e-Health створено для забезпечення переведення системи охорони здоров’я України в електронний формат, захисту прав лікарів і пацієнтів, інтеграції електронних медичних систем.

Усі ці заходи є будівельними блоками ідеї полегшення контролю за здоров’ям для широкої громадськості. А зараз, як бачимо, його реалізація значною мірою зустрічається, наприклад, у мобільних додатках, які доступні кожному власнику смартфона та у всесвітній павутині. Зокрема, є мобільні додатки, які допомагають пацієнтам правильно та вчасно приймати ліки.

1.2 Аналітичний огляд відомих технічних рішень контролю прийняття ліків

На сьогодні кількість різних мобільних додатків надзвичайно велика. Обираючи найкращий варто зважати на індивідуальні особливості людини та її потреби. Тому важливо розуміти, чим додатки відрізняються один від одного, який функціонал вам потрібен, а також не менш важливим критерієм є вартість.

У даному розділі наведено огляд найпопулярніших і найбільш розповсюджених додатків у світі: MediSafe App, Dosecast, CareClinic і Lady Pill Reminder.

MediSafe — це простий, зручний мобільний додаток, який пропонує ряд функцій, які полегшують керування рецептами. Додаток описується як «віртуальний дот», який надсилає користувачам персоналізовані нагадування про те, щоб приймати ліки відповідно до вказівок. Повідомлення також містять попередження про взаємодію з лікарськими засобами, щоб користувачі знали, чи не слід їм приймати ліки з певними ліками, їжею чи напоями [5].

Одним з недоліків програми MediSafe є те, що розробники надають медичним компаніям анонімні зведені дані користувачів. Хоча ці дані є анонімними і можуть бути корисними для розробки кращих методів лікування

та систем охорони здоров'я, деяким користувачам може не сподобатися той факт, що їхні дані збираються та передаються.

На певних пристроях нагадування про сповіщення MediSafe вимикаються, коли пристрій перебуває в беззвучному режимі, тому користувачі повинні залишити звук увімкненим, щоб отримати нагадування. Користувацький інтерфейс застосунку MediSafe зображено на рисунку 1.1.

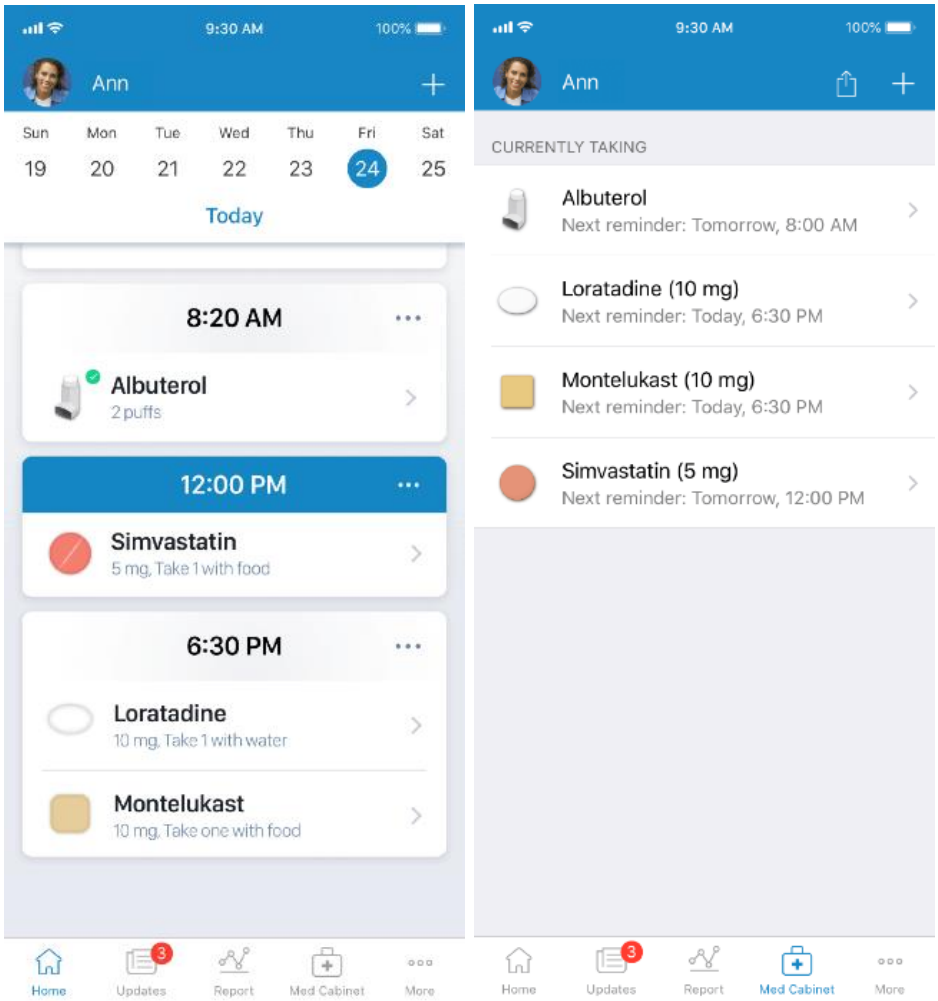

Рисунок 1.1 – Скріншоти роботи MediSafe

Існує безкоштовна версія DoseCast, але для користувачів, яким потрібна додаткова функціональність, є також платна версія Pro. Безкоштовні функції включають можливість встановлювати сповіщення про прийом ліків, вводити індивідуальні дозування, робити нотатки про окремі ліки та розумне

відключення звуку, тому сповіщення не надсилаються, коли користувачі сплять [6].

Найбільшим недоліком DoseCast є те, що деякі функції, як-от підрахунок таблеток і нагадування про поповнення, які включені безкоштовно в інші програми, доступні лише в платній Pro версії цієї програми.

Якщо користувачі протягом дня приймають різні дози одного і того ж препарату, вони повинні створити окреме нагадування для кожної дози. Користувацький інтерфейс застосунку DoseCast зображено на рисунку 1.2.

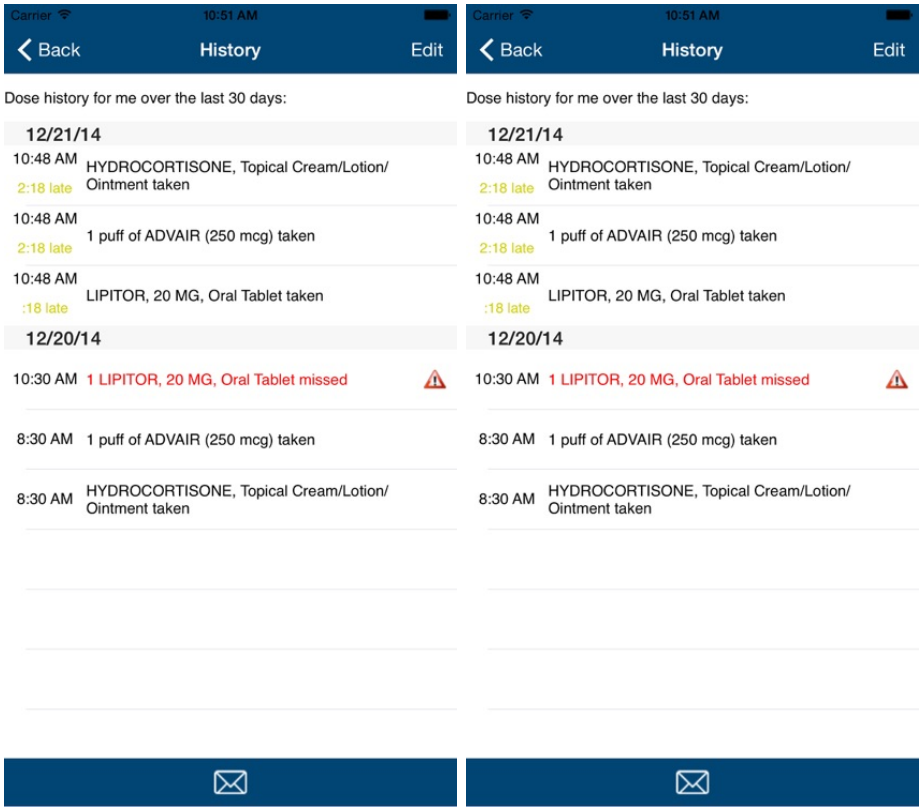

Рисунок 1.2 – Скріншоти роботи DoseCast

CareClinic — це комплексна програма для управління охороною здоров’я, яка полегшує пацієнтам не забувати приймати ліки, замовляти поповнення рецептів, записувати життєво важливі показники та стежити за призначеннями лікаря [7].

У додатку CareClinic користувачі можуть налаштувати нагадування про ліки в групах, для кількох ліків, що приймаються одночасно щодня, або

індивідуальні нагадування про ліки, які приймаються за унікальним графіком. У CareClinic є функція сканування, яка дозволяє користувачам легко додавати нові ліки до свого облікового запису.

Наразі програма CareClinic доступна лише англійською мовою, що обмежує її доступність лише для користувачів, які розмовляють англійською. Користувацький інтерфейс застосунку CareClinic зображено на рисунку 1.3.

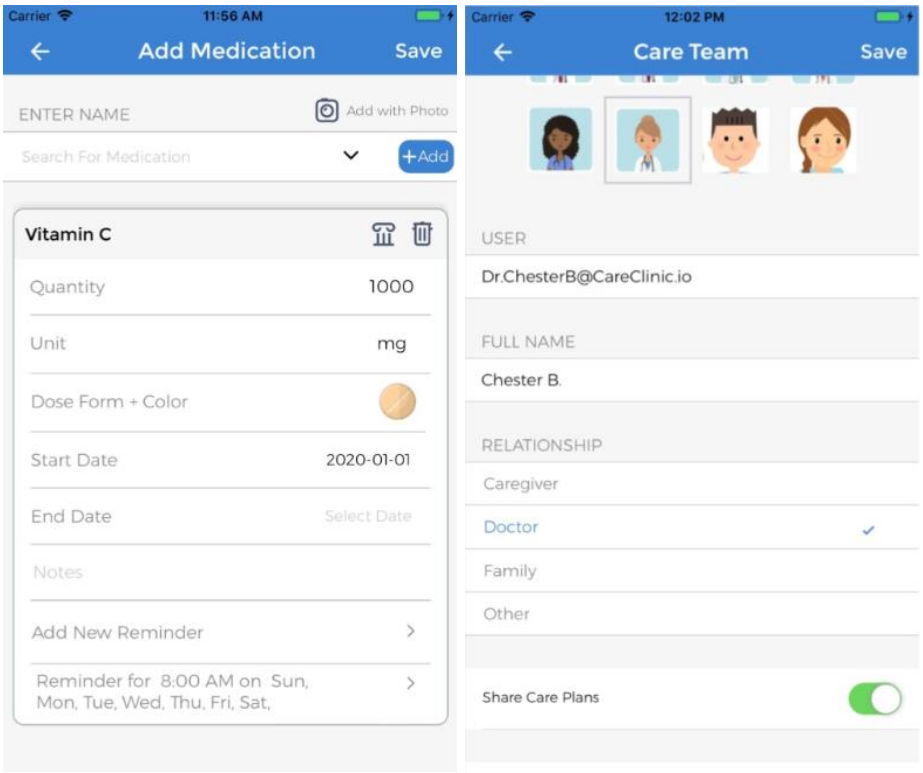

Рисунок 1.3 – Скріншоти роботи CareClinic

Жінки, які колись приймали протизаплідні засоби, знають, що їх ефективність багато в чому залежить від прийому таблеток за розкладом. Застосунок Lady Pill Reminder допомагає жінкам контролювати процес прийому ліків щодня і в один і той же час.

Головний мінус Lady Pill в тому, що наразі він доступний лише для пристроїв Android. Користувачам пристроїв Apple, яким потрібна програма такого типу, доведеться шукати в іншому місці. Хоча Lady Pill добре виконує своє призначення, його функціональні можливості обмежені за межами цієї

конкретної мети. Додавання додаткового функціоналу може зробити його більш вигідним для користувачів, які очікують більше від своїх програм [8]. Користувацький інтерфейс застосунку Lady Pill Reminder зображено на рисунку 1.4.

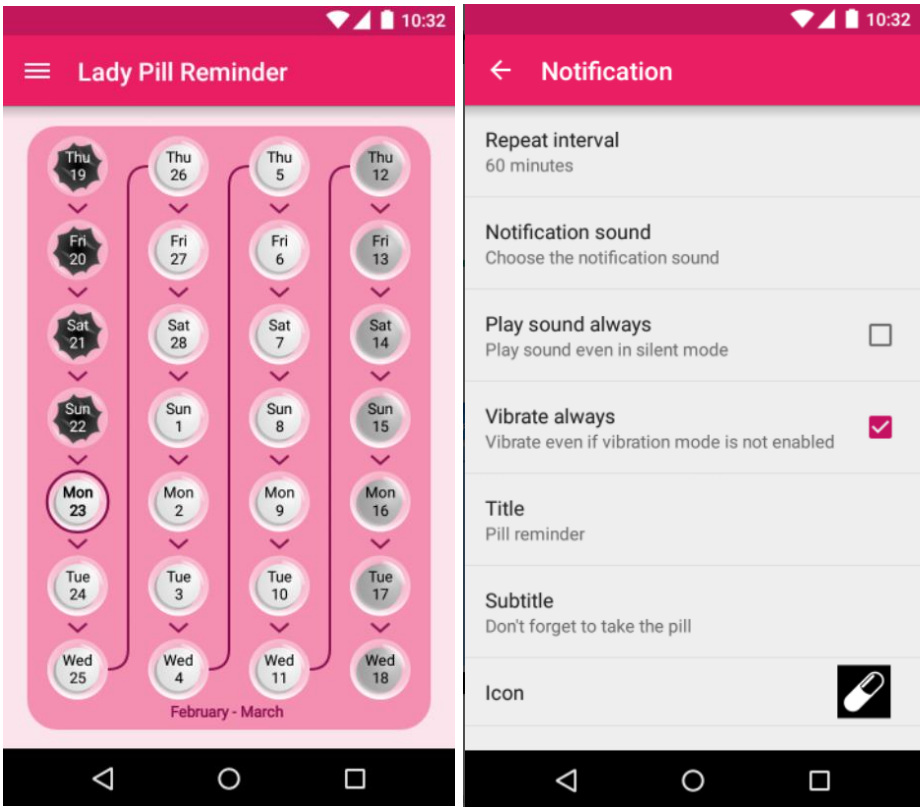

Рисунок 1.4 – Скріншоти роботи Lady Pill Reminder

На основі проведеної порівняльної характеристики сучасних засобів для прийому ліків можна зробити висновок, що є потреба в розробці програмного забезпечення, яке міститиме усі необхідні функції для виконання поставлених завдань і компенсує зазначені недоліки функціоналу існуючих засобів.

1.3 Аналіз методів контролю ліків

Управління та контроль прийняття ліків становлять критичну складову у забезпеченні ефективного та безпечного лікування пацієнтів. Неправильний прийом ліків може призвести до погіршення стану здоров'я, ускладнень та

навіть гірших результатів лікування. З цієї причини розробка інформаційних технологій для контролю прийняття ліків набула великого значення. У цьому розділі ми проведемо аналіз різних методів контролю прийняття ліків з акцентом на методах ведення інформації та нагадуванні пацієнтам.

Один із найпоширеніших та традиційних методів контролю прийняття ліків полягає в використанні паперових календарів та журналів, де пацієнти вручну записують дату та час прийому лікарських засобів. Цей метод є простим та доступним, але має певні обмеження.

Одним із головних недоліків цього методу є відсутність можливості надавати автоматичні нагадування. Пацієнт повинен самостійно слідкувати за графіком та пам'ятати про прийом ліків, що може бути особливо складним у випадках, коли необхідно приймати ліки регулярно і в певний час.

Деякі пацієнти використовують свого сімейного лікаря або медичний персонал для отримання порад щодо прийому ліків. Сімейні лікарі можуть надавати індивідуальні поради та спостерігати за дотриманням рецепту, але цей підхід обмежується годинами роботи медичного закладу та доступністю лікаря.

Незважаючи на корисність цього методу, він не завжди забезпечує постійний та ефективний контроль прийняття ліків. Пацієнти можуть опускати прийом ліків поза часом консультацій, а також мати обмежений доступ до лікаря у випадках, коли виникають питання або потребують додаткового контролю.

Мобільні додатки для контролю прийняття ліків стали дуже популярними завдяки їхній зручності та можливості надавати нагадування через сповіщення на смартфоні. Вони дозволяють пацієнтам вести журнал прийому ліків, а також зберігати інформацію про рецепти та дозування. Мобільні додатки зазвичай мають інтуїтивний інтерфейс та можуть надавати корисну інформацію про ліки та їх взаємодію.

Проте, важливо враховувати, що деякі пацієнти можуть бути не знайомі з сучасними технологіями або не мати доступу до смартфонів, що може обмежувати застосування цього методу.

Електронні системи для контролю прийняття ліків дозволяють пацієнтам вести журнал прийому ліків та надавати інформацію про рецепти та дозування. Вони також можуть надавати нагадування через електронну пошту або текстові повідомлення. Ці системи зазвичай мають більше можливостей щодо персоналізації та внесення змін у графік прийому ліків.

Після докладного аналізу різних методів контролю прийняття ліків, виділяється важливість розробки інформаційної системи, яка базується на веденні інформації про прийом ліків та нагадуванні пацієнтам про правильний прийом. Цей підхід об'єднує зручність мобільних додатків та доступність електронних систем, забезпечуючи надійний та ефективний контроль прийняття ліків.

Таким чином, розробка інформаційної системи контролю прийняття ліків, яка базується на веденні інформації та нагадуванні пацієнтам, має великий потенціал для покращення якості медичної допомоги та зменшення ризику неправильного прийому ліків пацієнтами. Ця система може бути ефективною у веденні журналу прийому ліків, надаванні нагадувань та спостереженні за дотриманням рецепту. Важливо також розглядати індивідуальні потреби пацієнтів та їх можливості використовувати сучасні технології, щоб забезпечити доступність та ефективність системи.

1.4 Постановка задачі

В сучасному світі зростання обчислювальної потужності та доступність великої кількості даних завдяки Інтернету створюють унікальні можливості для розробки інноваційних інформаційних технологій. Цей контекст дозволяє досліджувати нові підходи до вирішення важливих завдань, таких як контроль за вживанням ліків.

Інформаційна технологія контролю ліків має на меті стежити за прийомом лікарських препаратів користувачами. Вона може бути використана для забезпечення дотримання рекомендованого графіку прийому ліків, перевірки правильності дозування та попередження випадків пропусків лікування.

Ця інформаційна технологія має на меті підвищити ефективність лікування та забезпечити безпеку пацієнтів. Це особливо важливо в умовах зростаючого числа хронічних захворювань і необхідності у системі обліку та контролю прийому ліків.

На протязі цього дослідження ми зосередимося на розробці системи, яка буде спроможною відслідковувати режим прийому ліків у режимі реального часу. Крім того, ця система матиме можливість надавати інформацію не лише про одного користувача, а й враховуватиме потреби різних пацієнтів.

Наша інформаційна технологія має бути досить універсальною та налаштовуваною для різних груп пацієнтів та їх конкретних потреб.

Модуль, який буде розроблений в рамках цього дослідження, має надавати інтерфейс для користувачів на системах Android та IOS. Цей інтерфейс повинен бути зручним та легким у використанні, щоб користувачі могли ефективно взаємодіяти з системою.

Під час тренування моделі для відслідковування прийому ліків, великий обсяг даних є важливим фактором для досягнення високої точності. Оскільки створення власної бази даних може бути затратним та часоємким завданням, наша інформаційна технологія повинна надавати доступ до набору даних, включеного в комплект поставки програмного продукту.

Для підтвердження наукової актуальності та ефективності цієї інформаційної технології, необхідно розробити модуль для автоматизованого тестування її роботи на великому обсязі структурованих даних. Додатково, важливо розробити можливість валідації, яка дозволить проводити тестування на невідформатованих наборах даних.

У підсумку, створення цієї інформаційної технології дозволить вдосконалити контроль за вживанням ліків та забезпечити більшу ефективність та безпеку в лікувальних процедурах пацієнтів.

1.5 Висновок

Виконано аналіз предметної області та вивчено основні методи контролю за прийняттям ліків. У ході аналізу було встановлено, що використання інформаційної технології на основі експертних систем може забезпечити високу точність контролю прийняття ліків. Що є критично важливим для впровадження системи контролю прийняття ліків.

Здійснено огляд та аналіз існуючих програмних рішень, що призначені для використання у сфері контролю прийняття ліків. Виявлено, що сучасні програми не забезпечують достатню точність контролю прийняття ліків.

З цього аналізу стає очевидним, що існує потреба у розробці інформаційної системи, яка представлятиме собою програмний додаток з розширеними функціональними можливостями на основі моделі, здатної до високоякісного контролю прийняття ліків. (отримання зворотнього зв’язку та його аналіз)

Підсумовуючи, розробка цієї інформаційної технології в області контролю прийняття ліків є актуальною і може відкрити нові можливості для покращення якості медичних послуг та підвищення ефективності лікування пацієнтів.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ КОНТРОЛЮ ПРИЙНЯТТЯ ЛІКІВ

2.1 Розробка математичної моделі для інформаційної технології контролю прийняття ліків

Первиною метою інформаційної технології контролю прийняття ліків є розширення функціональних можливостей у відстеженні прийому медикаментів, а також визначати невідповідності у прийомі ліків, щоб своєчасно надавати повідомлення користувачам. При розробці математичної моделі нам слід враховувати оцінку самопочуття користувача, зміну його стану, та ризики.

Модель визначення композитного показника добробуту можна подати в наступній формулі 2.1:

$$CWS = w_1 \times Q_1 + w_2 \times Q_2 + \cdots + w_n \times Q_n, \quad (2.1)$$

де CWS – зведений показник добробуту. Зведений показник добробуту – це спосіб кількісної оцінки загального добробуту користувача на основі його відповідей на набір запитань. Кожне запитання вимірює певний аспект благополуччя, такий як фізичне здоров’я, психічне здоров’я, сон. Ідея полягає в тому, щоб об’єднати ці окремі показники в єдиний комплексний бал, який відображає загальний стан користувача. Q – оцінка питання та w це вага питання.

Аналіз змін у самопочутті пацієнта з часом має вирішальне значення для розуміння прогресування стану його здоров’я, впливу лікування та прийняття обґрунтованих клінічних рішень. У контексті зведеного показника добробуту це передбачає відстеження того, як цей показник змінюється в різні моменти часу та описується формулою 2.2:

$$RC = \frac{Current\ Scope - Previous\ Scope}{Time\ Interval}, \quad (2.2)$$

де RC – це оцінка зміни стану користувача за певний проміжок часу

Оцінка ризику на основі показників благополуччя передбачає аналіз цих показників для прогнозування ймовірності несприятливих подій або ускладнень. Цей процес допомагає виявити пацієнтів, які можуть мати підвищений ризик і потребують більш інтенсивного моніторингу або втручання. Оцінка зазвичай об'єднує дані про самопочуття з іншою відповідною клінічною інформацією, щоб отримати комплексний профіль ризику, який розраховується за формулою 2.3:

$$RS = a \times CWS + b \times MHS + c \times LFS, \quad (2.3)$$

RS – оцінка ризику, яка залежить від зведеного показника добробуту CWS, хронічних захворювань MHS та оцінки фактору способу життя LFS.

Кореляційний аналіз між показниками благополуччя та прихильністю до лікування є критично важливим аспектом розуміння та покращення результатів здоров'я пацієнтів. Прихильність до лікування, тобто ступінь правильного дотримання пацієнтом медичних рекомендацій щодо часу, дозування та частоти прийому ліків, може значно вплинути на його загальне самопочуття. Аналіз зв'язку між цими двома факторами може дати цінну інформацію для постачальників медичних послуг, який рахується за формулою 2.4.

$$r = \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{(n\sum x^2 - (\sum x)^2)(n\sum y^2 - (\sum y)^2)}}, \quad (2.4)$$

де x та y – індивідуальні показники благополуччя та значення прихильності до лікування відповідно. n – кількість точок даних.

Позитивна кореляція вказує на те, що з покращенням прихильності до лікування покращується і самопочуття пацієнта.

Негативна кореляція свідчить про те, що більш висока прихильність до лікування пов’язана з нижчими оцінками добробуту, що може вказувати на такі проблеми, як побічні ефекти ліків.

Кореляція, близька до нуля, означає, що між двома змінними практично немає прямого зв’язку.

За допомогою коефіцієнту кореляції Пірсона можна визначити силу лінійної залежності між змінними, інші види взаємозв'язків виявляються методами регресійного аналізу. Коефіцієнт кореляції може надати цінну інформацію у сфері ІТ охорони здоров’я, особливо для систем, призначених для моніторингу та підвищення прихильності до лікування. Це допомагає зрозуміти вплив прихильності до лікування на результати пацієнтів, тим самим інформуючи про стратегії залучення пацієнтів та управління здоров’ям. Дуже важливою для отримання надійних та статистично обґрунтованих результатів є оцінка значущості статистичних показників. Це цілий комплекс математичних процедур, що дають змогу відповісти на низку запитань щодо обчислених статистичних показників та параметрів вибіркової сукупності. Так, якщо ми обчислили коефіцієнти кореляції між двома ознаками й отримали число, що не дорівнює нулю, нас має зацікавити, чи справді цей коефіцієнт істотно відрізняється від нуля (а отже, фіксує наявність лінійного кореляційного зв'язку), чи ця різниця випадкова і спричинена лише похибкою нашої вибірки.

Кореляційний аналіз між показниками благополуччя та прихильністю до лікування дає важливу інформацію для покращення догляду за пацієнтами. Розуміння цього зв’язку допомагає визначити сфери, де потрібне втручання, будь то лікування ліками, навчання пацієнтів або системи підтримки. Цей аналіз має бути безперервним процесом, адаптованим до унікального шляху здоров’я кожного пацієнта.

2.2 Розробка структури роботи інформаційної системи контролю прийняття ліків

Аналізуючи вхідні дані функціоналу інформаційної системи контролю прийняття ліків, та досліджуючи дані, які потрібно зберігати, розроблено структуру інформаційної системи контролю прийняття ліків, яка зображена на рисунку 2.1.

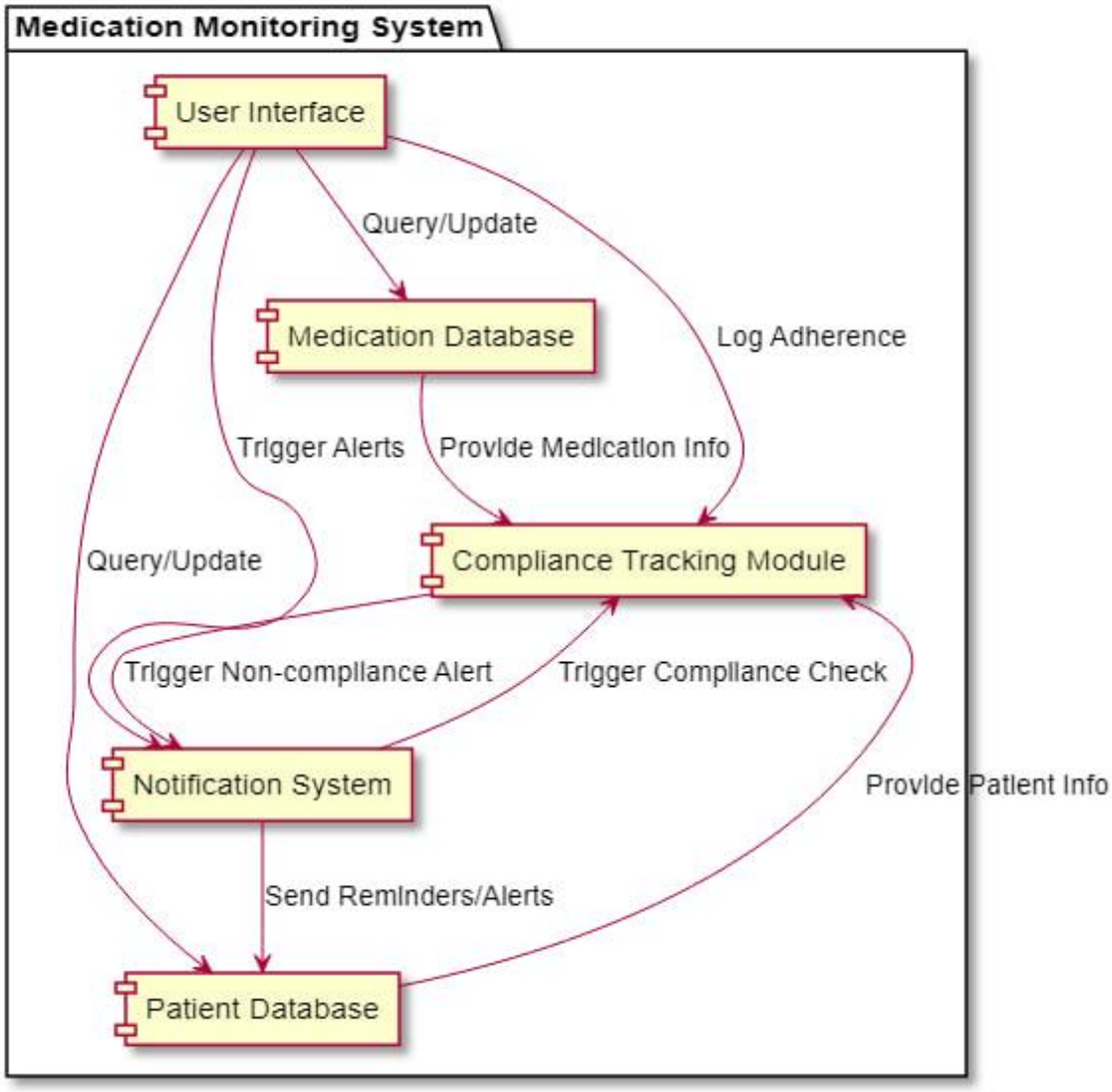

Рисунок 2.1− Структурна схема системи контролю прийому ліків

Структурна схема для інформаційної системи включає в себе такі елементи:

- Інтерфейс користувача – виступає в якості вхідної точки в систему, через який користувачі (пацієнти) взаємодіють із системою. Цей модуль дозволяє користувачам запитувати та оновлювати інформацію як у базах даних ліків, так і в базах даних пацієнтів, активовує сповіщення через систему сповіщень та реєструє дані про прихильність до лікування в модулі відстеження відповідності.
- База даних ліків – зберігає детальну інформацію про ліки, включаючи назви, дозування та графіки. Надає інформацію про ліки до модуля відстеження відповідності, щоб допомогти контролювати дотримання.
- База даних пацієнтів – містить інформацію про пацієнта, таку як історія хвороби, поточні ліки та особисті дані. Надає інформацію про пацієнта до модуля відстеження відповідності.
- Система сповіщень – керує надсиланням нагадувань і сповіщень пацієнтам щодо графіків прийому ліків. Надсилає нагадування та сповіщення до бази даних пацієнтів та запускає перевірки відповідності в модулі відстеження відповідності.
- Модуль відстеження відповідності – контролює та реєструє дотримання пацієнтами ліків, перевіряючи, чи пацієнти приймають ліки відповідно до призначень. Отримує інформацію про ліки з бази даних про ліки та інформацію про пацієнта з бази даних пацієнтів. Повідомляє Систему сповіщень про випадки невідповідності.

Ці модулі є тісно пов’язані між собою та робота одних залежить від роботи інших:

- Інтерфейс користувача та бази даних – користувачі взаємодіють з інтерфейсом користувача, щоб отримати доступ до інформації в базах даних ліків і пацієнтів та оновити її, що робить інтерфейс користувача важливим каналом для потоку інформації.
- Інтерфейс користувача та система сповіщень – інтерфейс користувача запускає сповіщення в системі сповіщень, що дозволяє проактивно залучати пацієнтів і відсилати нагадування.

- Інтерфейс користувача та відстеження відповідності – за допомогою інтерфейсу користувача користувачі можуть реєструвати прихильність до лікування, що має вирішальне значення для модуля відповідності щоб відстежувати та аналізувати дотримання пацієнтом режиму.
- База даних ліків і модуль відповідності – модуль відстеження відповідності покладається на точну інформацію про ліки з MedDB, щоб контролювати, чи пацієнти дотримуються призначених схем прийому ліків.
- База даних пацієнтів – інформація про пацієнта з PatDB допомагає модулю відповідності персоналізувати моніторинг прихильності.
- Роль системи сповіщень – система сповіщень діє як міст між базами даних (MedDB, PatDB) і модулем відповідності, полегшуючи потік нагадувань і попереджень на основі графіків прийому ліків і дотримання пацієнтом режиму.
- Система відстеження відповідності та сповіщення – модуль відповідності взаємодіє з системою сповіщень, щоб ініціювати сповіщення у випадках невідповідності, забезпечуючи своєчасне втручання.

Система моніторингу ліків розроблена для роботи як інтегрована одиниця, де кожен модуль відіграє певну роль у забезпеченні ефективного управління ліками. Взаємодія між цими модулями забезпечує безперебійний потік інформації, необхідний для точного та ефективного моніторингу прийому ліків. Інтерфейс користувача служить для користувача шлюзом до системи, бази даних (MedDB, PatDB) діють як сховища важливої інформації, система сповіщень керує спілкуванням з пацієнтами, а модуль відповідності є ключовим для відстеження та забезпечення прихильності до лікування.

На рисунку 2.2 наведено діаграму діяльності роботи системи враховуючи взаємодію описаних сутностей.

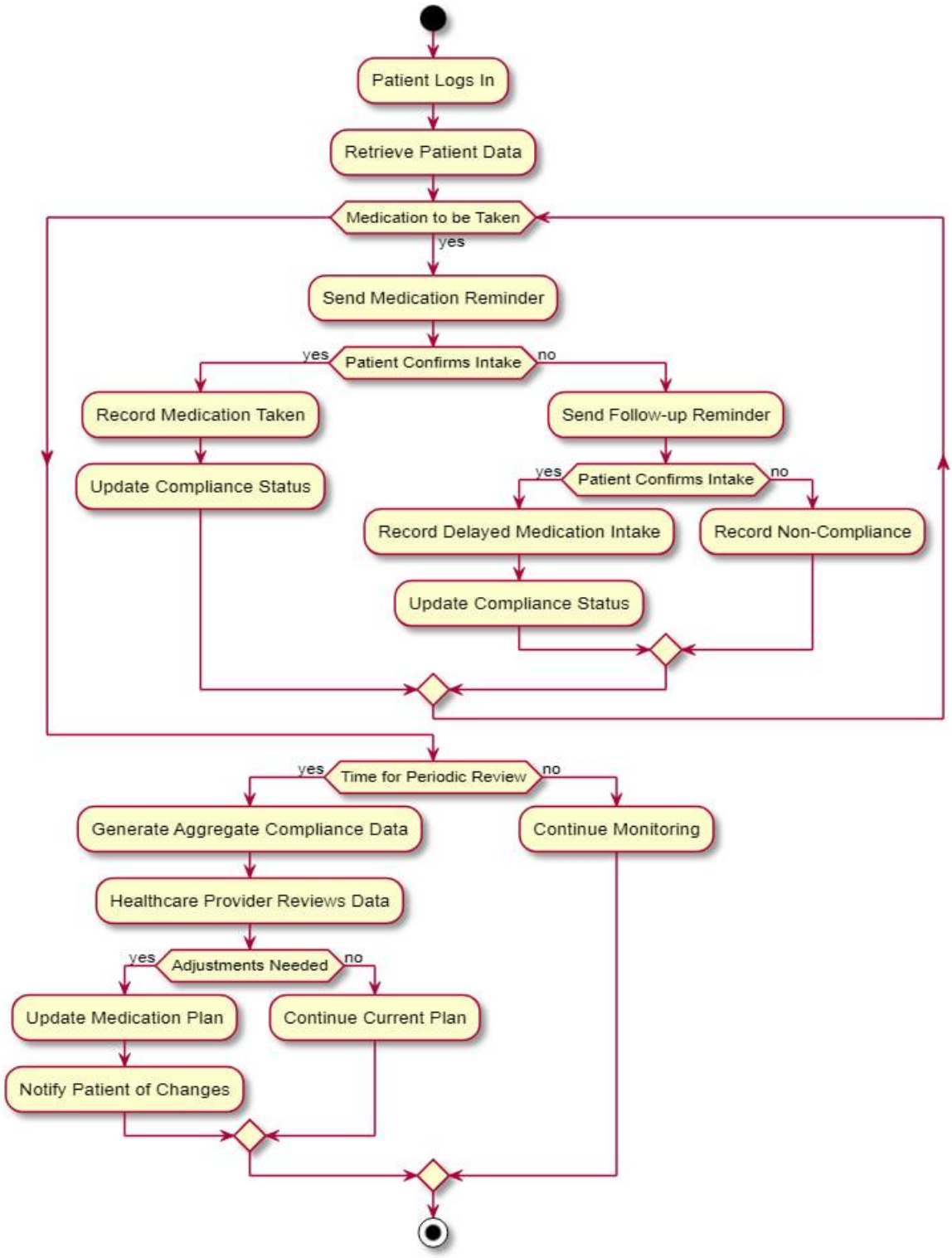

Рисунок 2.2 – Діаграма діяльності роботи програмного забезпечення

Як показано на діаграмі діяльності система контролю прийняття ліків, працює послідовно та динамічно, щоб контролювати дотримання пацієнтом споживання ліків. Процес починається, коли пацієнт входить в систему, ініціюючи отримання персональних даних про ліки. Цей крок важливий,

оскільки він гарантує, що нагадування та наступні дії адаптовані до конкретного режиму прийому ліків.

Після входу пацієнта в систему та отримання доступу до його даних система переходить у фазу безперервного моніторингу. На цьому етапі він перевіряє, чи потрібно приймати будь-які ліки в поточний час. Якщо пацієнту час прийняти ліки, система надсилає нагадування. Реакція пацієнта на це нагадування має вирішальне значення для визначення наступних кроків. Якщо пацієнт підтверджує, що приймав ліки, система фіксує це споживання та оновлює статус відповідності, вказуючи на дотримання призначеного режиму.

Однак, якщо пацієнт не підтверджує прийом, система реагує, надсилаючи наступне нагадування, надаючи пацієнту ще одну можливість дотримуватися графіка прийому ліків. Реакція пацієнта на це подальше нагадування знову є ключовою. Якщо пацієнт підтверджує прийом на цьому етапі, система реєструє це як відкладений прийом і відповідно оновлює статус відповідності. З іншого боку, якщо пацієнт все ще не підтверджує прийом ліків, система реєструє цю подію як невідповідність.

Крім цих дій, заснованих на негайному реагуванні, система також містить механізм періодичного перегляду. Через заздалегідь визначені проміжки часу система генерує зведені дані про прихильність пацієнта до лікування, які потім переглядає медичний працівник. Цей огляд є ключовим аспектом системи, що дозволяє здійснювати професійний нагляд і втручання. Під час цього перегляду, якщо лікар визначає, що необхідні коригування плану прийому ліків, система оновлює план і сповіщає пацієнта про ці зміни, гарантуючи, що режим лікування залишається ефективним і оновленим. У випадках, коли зміни не потрібні, система продовжує працювати за поточним планом.

Функціонування системи характеризується циклічністю та оперативним характером. Вона постійно контролює прийом ліків, реагує на дії пацієнта нагадуваннями та спостереженнями, а також адаптується до змін у плані прийому ліків відповідно до вказівок лікаря. Цей динамічний підхід гарантує,

що система продовжує працювати з потребами пацієнта, і забезпечує надійну основу для ефективного прийому ліків.

На рисунку 2.3 наведено UML діаграму класів, на ній зображено зв’язки між класами та структурами поекту.

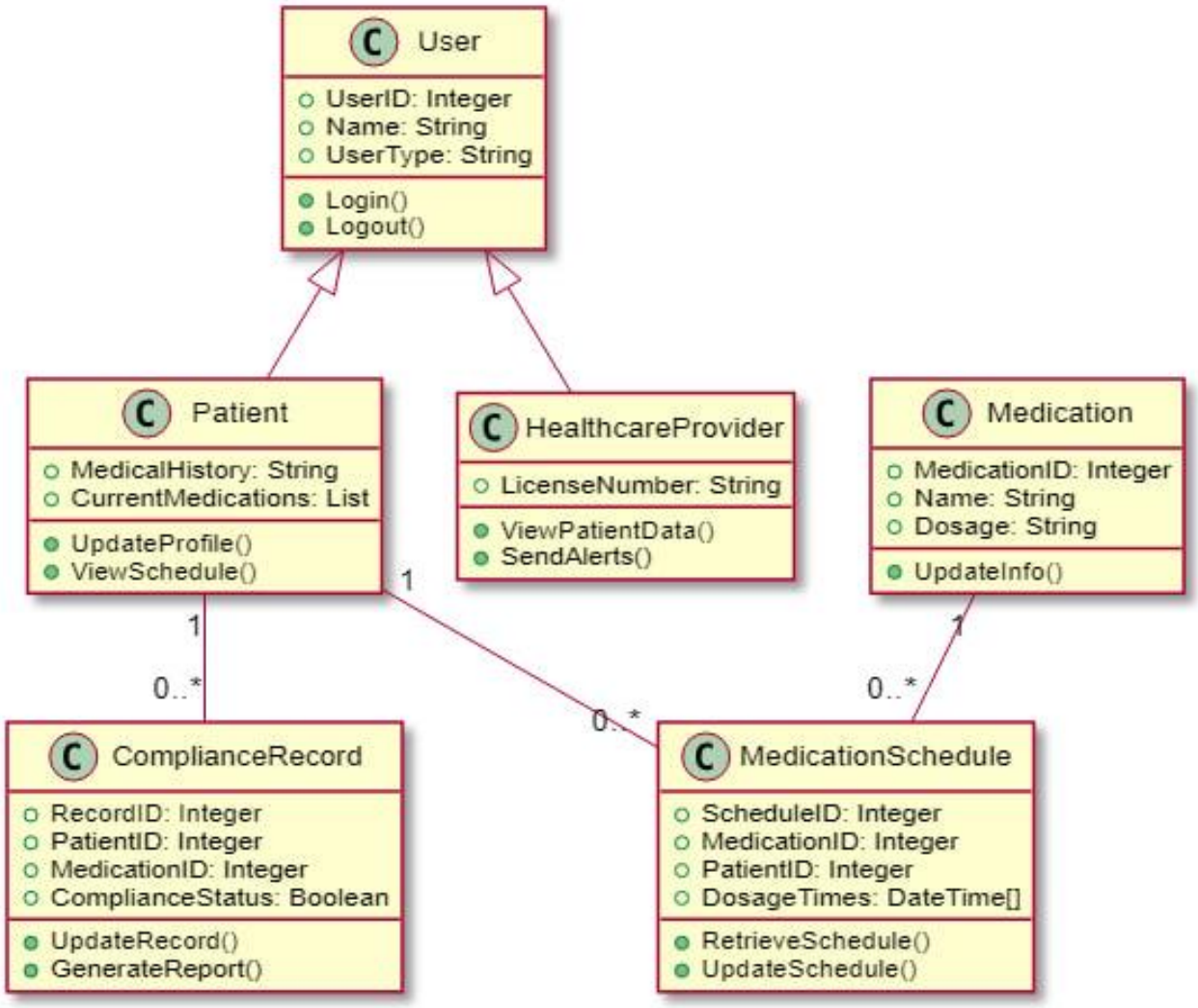

Рисунок 2.3 – UML-діаграма класів

Для системи контролю прийняття ліків реалізовано наступні класи:

- Користувач – представляє загального користувача системи, охоплюючи основні атрибути та поведінку, загальні для всіх типів користувачів. Атрибути включають унікальний ідентифікатор користувача, ім’я та тип. Методи включають такі основні функції, як вхід і вихід.

- Пацієнт (підклас користувача) – спеціалізований тип Користувача, що представляє пацієнтів, які використовують систему. Успадковує атрибути та методи від класу User та має додаткові атрибути, такі як історія хвороби та поточні ліки. Включає спеціальні методи, як-от оновлення профілю та перегляд розкладу прийому ліків.
- Постачальник медичних послуг (підклас користувача) – інший спеціалізований тип Користувача, що представляє медичних працівників. Успадковує базову структуру класу User та містить нові атрибути, такі як номер ліцензії. Включає методи перегляду даних пацієнтів і надсилання сповіщень.
- Ліки – представляє інформацію, пов’язану з ліками. Атрибути включають ідентифікатор препарату, назву та відомості про дозування, а також містить методи оновлення інформації про ліки.
- Розклад прийому ліків – відноситься до планування прийому ліків для пацієнтів. Атрибути включають ідентифікатор розкладу, ідентифікатор препарату (посилання на клас ліків), ідентифікатор пацієнта (посилання на клас пацієнта) і час дозування. Включає методи для отримання та оновлення графіків прийому ліків.
- Запис про відповідність – відстежує дотримання пацієнтами режиму прийому ліків. Атрибути включають ідентифікатор запису, ідентифікатор пацієнта, ідентифікатор ліків (створення зв’язку з класами пацієнтів і ліків) і статус відповідності. Містить методи для оновлення запису та створення звітів про відповідність.

У системі контролю прийняття ліків, як це визначено діаграмою UML, зв’язки між різними об’єктами утворюють цілісну та структуровану мережу, яка лежить в основі функціональності системи.

На базовому рівні клас User служить наріжним каменем, інкапсулюючи основні атрибути та функціональні можливості, спільні для всіх типів користувачів системи. З цієї основи виникають два спеціалізовані підкласи: пацієнт і медичний працівник. Ці підкласи успадковують основні властивості

класу User, а також представляють унікальні атрибути та методи, які відповідають їхнім конкретним ролям у системі. Цей зв'язок успадкування гарантує, що як пацієнти, так і постачальники медичних послуг, незважаючи на їхні відмінні функції, мають спільну структурну та функціональну основу, сприяючи послідовності та ефективності дизайну системи.

Клас MedicationSchedule утворює важливу асоціацію з класами Medication і Patient. Цей зв'язок має ключове значення, оскільки він напряму пов'язує режим лікування кожного пацієнта з конкретними ліками, забезпечуючи точне узгодження графіків як з деталями ліків, так і з індивідуальними потребами пацієнтів. Таким чином, клас MedicationSchedule стає центральним вузлом, який інтегрує та координує інформацію, пов'язану з ліками, що стосується догляду за кожним пацієнтом.

Доповнює цю структуру клас ComplianceRecord, який тісно пов'язаний з класом Patient. Кожен запис про відповідність є розширенням профілю пацієнта, що пропонує детальну інформацію про дотримання ним призначених схем лікування. Крім того, клас ComplianceRecord також асоціюється з класом Medication, підкреслюючи важливість відстеження відповідності щодо конкретних ліків. Цей подвійний зв'язок не тільки полегшує комплексний моніторинг прихильності до лікування, але й дає змогу системі адаптувати свою реакцію та втручання на основі як факторів, що стосуються конкретного пацієнта, так і характеристик ліків.

Загалом, система контролю прийняття ліків, як зображено на діаграмі UML, характеризується гармонійним поєднанням ієрархічних та асоціативних зв'язків. Ці зв'язки гарантують, що система може ефективно керувати та відстежувати графіки прийому ліків і дотримання, а клас користувача забезпечує уніфіковану базу для більш спеціалізованих функцій пацієнта та медичного працівника. Взаємопов'язаний характер класів Medication, MedicationSchedule та ComplianceRecord ще більше підсилює здатність системи зберігати точні та актуальні записи, які є важливими для ефективного управління ліками та догляду за пацієнтами.

2.3 Проектування структури бази даних для інформаційної технології контролю прийняття ліків

Для ефективної роботи системи контролю прийняття ліків важливо забезпечити швидку обробку великих обсягів даних. Ці дані мають бути розподілені між кількома серверами для підвищення продуктивності та безпеки. Одним з оптимальних рішень для керування цією складною архітектурою даних є використання Microsoft SQL Server [9], провідної системи управління реляційними базами даних (СУБД) на ринку, відомої своєю надійністю в середовищі бізнес-аналітики та корпоративного аналізу. Microsoft SQL Server особливо підходить для зберігання важливої інформації в реляційних базах даних і для ефективного керування даними завдяки зручному візуальному інтерфейсу, комплексним налаштуванням і універсальним інструментам. Це особливо важливо для систем з реєстрацією користувачів і можливостями входу. Побудований на мові Transact-SQL [10], яка включає набір програмних розширень, Microsoft SQL Server можна адаптувати як для локальних, так і для хмарних програм.

Основні переваги використання Microsoft SQL Server:

1. Покращена безпека даних. Microsoft SQL Server розроблено для підтримки надійної безпеки бази даних. Він забезпечує безпечну обробку даних, надаючи контрольований доступ до таблиць і обмежуючи можливості редагування. Ця функція має вирішальне значення для захисту конфіденційних даних пацієнтів і ліків.
2. Простота конфігурації. Популярність СУБД частково пояснюється її зручним налаштуванням. Його інтуїтивно зрозуміла студія керування забезпечує безперебійну взаємодію з базою даних, що робить її доступною навіть для тих, хто має обмежені технічні знання.
3. Ефективна обробка даних. Microsoft SQL Server дозволяє індексувати дані, що має вирішальне значення для швидкого пошуку даних.

Індексація створює структуру на основі диска, пов’язану з таблицями або представленнями, підвищуючи швидкість операцій пошуку.

Студія управління пропонує інтуїтивно зрозумілий спосіб роботи з базою даних. Він підтримує звітність і аналітику, необхідні для моніторингу прихильності до лікування та результатів пацієнтів.

Як показано на рисунку 2.4, база даних ретельно організована із зв’язками між об’єктами, такими як ліки, розклад та користувачі.

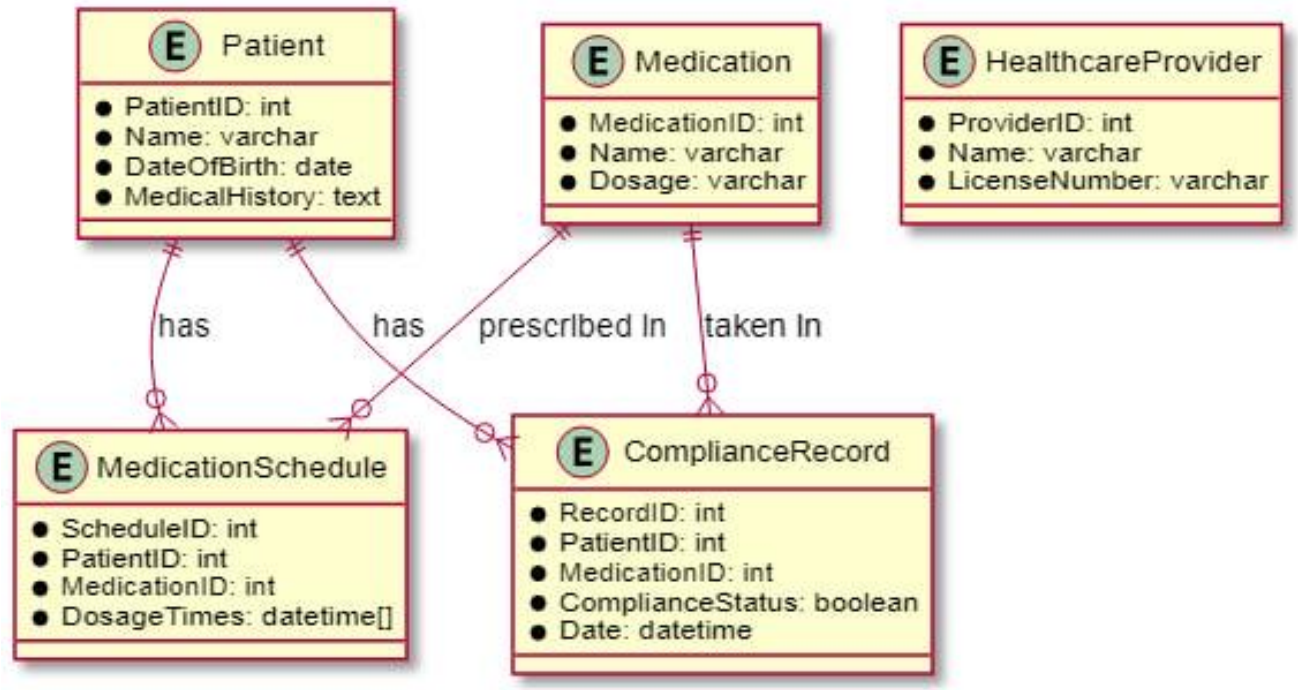

Рисунок 2.4 – Загальна схема бази даних роботи системи

В основі цієї структури лежить таблиця пацієнтів, яка зберігає важливу інформацію про пацієнтів, включаючи їхні унікальні ідентифікатори, імена, дати народження та повну історію хвороби. Ця таблиця має вирішальне значення для ведення індивідуальних профілів пацієнтів і є основою для персоналізованого управління ліками.

Підхід, орієнтований на пацієнта, доповнює таблиця HealthcareProvider. Ця таблиця призначена для зберігання деталей про медичних працівників, включаючи їхні унікальні ідентифікатори, імена та номери ліцензій. Він

відіграє ключову роль у автентифікації постачальників медичних послуг і зв’язуванні їх із конкретними видами діяльності з догляду за пацієнтами в системі.

Центральне місце в аспекті управління ліками в системі займає таблиця ліків. Ця таблиця містить детальну інформацію про різні ліки, включаючи їхні унікальні ідентифікатори, назви та інструкції щодо дозування. Він служить основним сховищем усіх ліків, призначених у системі, забезпечуючи точність і легкий доступ до даних про ліки.

Таблиця MedicationSchedule доповнює розрив між пацієнтами та їхніми схемами лікування. У цій таблиці наведено схему прийому ліків для кожного пацієнта, пов’язуючи пацієнтів із відповідними ліками. Він містить інформацію про час і дозування ліків, що відіграє вирішальну роль у забезпеченні дотримання призначених графіків прийому ліків.

Нарешті, таблиця ComplianceRecord призначена для моніторингу та реєстрації дотримання пацієнтами графіків прийому ліків. Документуючи кожен випадок прийому ліків і пов’язуючи його з відповідним пацієнтом і ліками, ця таблиця надає комплексне уявлення про поведінку пацієнта при прийомі ліків. Він записує, чи дотримуються пацієнти призначених режимів, що є життєво важливим для моніторингу та покращення результатів здоров’я пацієнтів.

Зв'язки між цими таблицями вдумливо встановлені. Таблиця «Пацієнт» пов’язана з таблицями MedicationSchedule і ComplianceRecord, що вказує на те, що кожен пацієнт може мати кілька графіків прийому ліків і серію записів відповідності. Цей зв’язок відображає здатність системи відстежувати графік прийому ліків кожним пацієнтом і його дотримання з часом. Подібним чином таблиця «Лікарські засоби» пов’язана з таблицями «Розклад ліків» і «Запис відповідності», демонструючи, що кожен препарат може відображатися в різних розкладах і записах відповідності.

Підводячи підсумок, можна сказати, що структура бази даних системи медичних інформаційних технологій є комплексною структурою, яка об’єднує

пацієнтів, постачальників медичних послуг, ліки та дані про прихильність. Стимулюючи ці зв’язки, система забезпечує злагоджений та ефективний підхід до управління ліками, значно сприяючи покращенню догляду за пацієнтами та точності введення ліків.

Дизайн бази даних застосовує нормалізацію до третьої нормальної форми, гарантуючи, що кожна таблиця містить скалярні дані, щоб уникнути повторюваних кортежів. Друга нормальна форма досягається, оскільки кожен неключовий атрибут залежить від первинного ключа, а третя нормальна форма усуває транзитивні залежності в таблицях.

Впровадження Microsoft SQL Server у систему інформаційних технологій контролю прийняття ліків забезпечує значні переваги з точки зору безпеки, простоти використання та ефективності обробки даних. Його надійна структура в поєднанні з розширеними функціями, такими як індексація та комплексна нормалізація, робить його ідеальним вибором для керування складними даними охорони здоров’я, гарантуючи, що система залишається чутливою, безпечною та надійною.

2.4 Розробка експертної системи для контролю прийняття ліків

Експертні системи представляють важливу галузь штучного інтелекту, яку часто згадують як яскраві приклади штучних когнітивних систем (ACS). Ці системи, по суті, є спеціалізованими інструментами, розробленими для узгодження з конкретними потребами та стандартами конкретної області чи галузі експертних знань. Ядро експертної системи можна уявити як програмну структуру, яка полегшує представлення знань і забезпечує механізми для логічного висновку [11].

Термін «оболонка» в контексті експертних систем позначає компонент програмного забезпечення, який охоплює інтерфейс, механізм логічного мислення та структурований підхід до керування базою знань разом із відповідними методами представлення цих знань. Для неексперта це можна

порівняти з порожньою посудиною, яка очікує вливання експертних знань. Після отримання цих знань у поєднанні з механізмом інтерфейсу система стає здатною обробляти запити користувачів і розробляти рішення для конкретних завдань користувача.

По суті, будь-яка комп’ютерна програма, яка, оснащена відповідною базою знань, перетворюється на експертну систему, називається оболонкою експертної системи. Це сховище як фактичних, так і евристичних знань. Інструменти, які використовуються в експертних системах, пропонують одну або більше структур для представлення знань, що стосуються області застосування. Деякі системи можуть інтегрувати фрейми (об’єкти) і правила IF-THEN для цієї мети. Наприклад, у PROLOG знання формулюються у формі логічних пропозицій.

Механізми логічного висновку в цих системах мають завдання маніпулювати символічною інформацією та знаннями в базі знань, створюючи таким чином логічну послідовність для вирішення проблеми. Ці механізми можуть варіюватися від простого modus ponens, який є основною формою логічного аргументу, до більш складних структур, таких як зворотне ланцюжок правил ЯКЩО-ТОДІ, або навіть міркування на основі конкретних випадків, що особливо важливо у сфері контролю над ліками, де кожен Випадок може представляти унікальні змінні та потребувати індивідуальних рішень. Загальна схема роботи експертної системи зображена на рисунку 2.5.

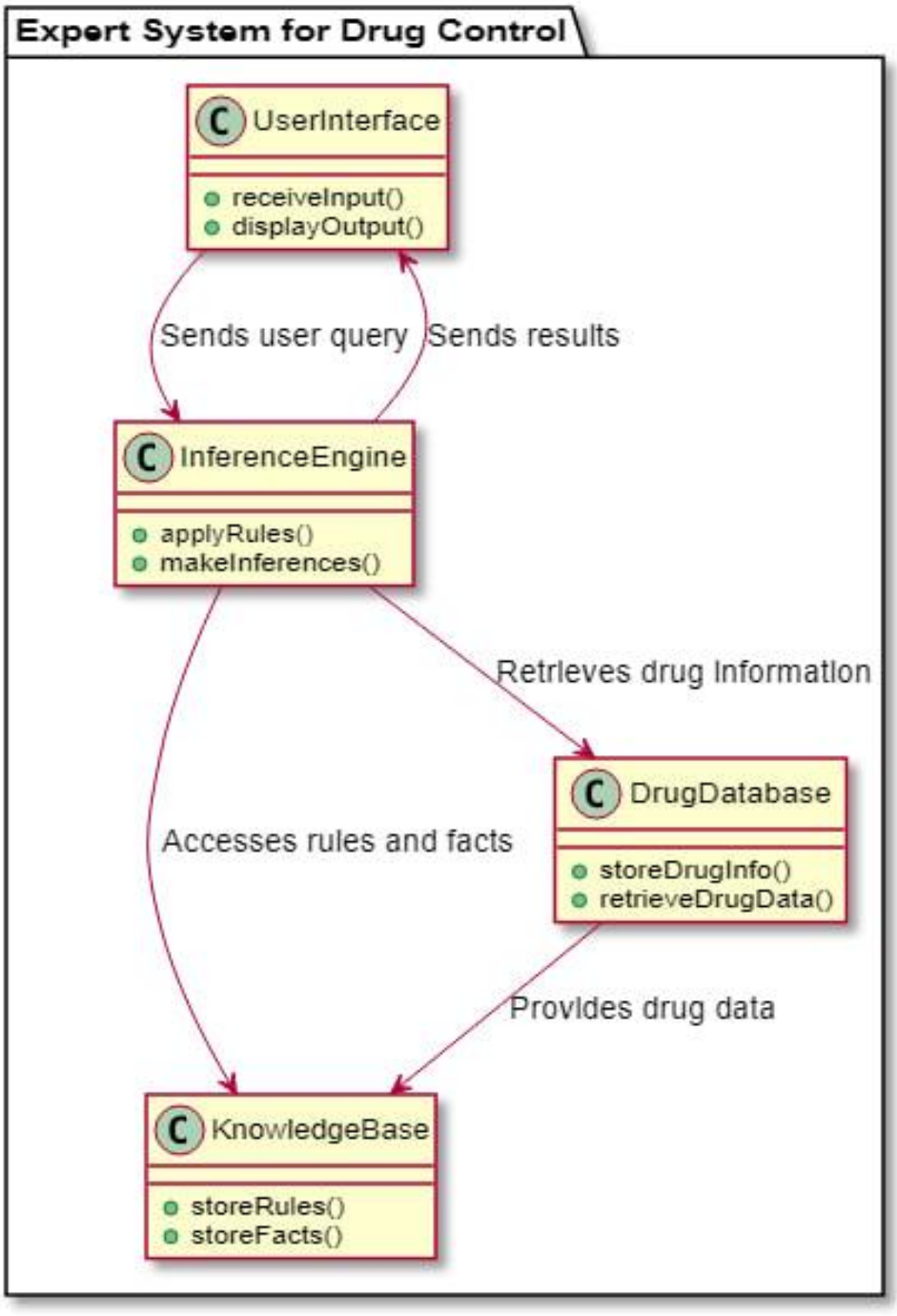

Рисунок 2.5 – Загальна схема роботи експертної системи

Входом в систему є інтерфейс користувача, який є основною точкою взаємодії для користувачів. Цей інтерфейс створено для захоплення запитів користувачів і відображення результатів обробки системою. Він діє як шлюз, через який передаються запити та отримуються результати, що робить його важливим каналом потоку інформації.

Центральним елементом експертної системи є Inference Engine. Цей основний компонент застосовує логічні правила до зібраної інформації та використовує свої можливості міркування, щоб робити висновки або робити прогнози. Це серце системи, де відбувається фактична обробка даних. Інференційний механізм отримує дані з бази знань, яка зберігає велику кількість правил і фактів, що стосуються контролю прийняття ліків. Ця база даних може містити вичерпну інформацію про вказівки щодо ліків, взаємодії, побічні ефекти та нормативні стандарти. Точність і глибина бази знань мають вирішальне значення, оскільки вони безпосередньо впливають на ефективність аналізу й висновків механізму логічного висновку.

Крім того, система містить спеціалізовану базу даних про лікарські засоби. Цей компонент є сховищем детальної та актуальної інформації про різні препарати. Він надає важливі дані, такі як використання ліків, взаємодія, протипоказання та інші важливі деталі. Інференційний механізм отримує доступ до цієї бази даних для отримання інформації про конкретні ліки в рамках процесу прийняття рішень. Ця інтеграція гарантує, що висновки механізму базуються на найновіших і повних доступних даних про ліки.

Робота системи характеризується безперебійною взаємодією між цими компонентами. Запити користувачів, ініційовані через інтерфейс користувача, обробляються механізмом висновків, який потім взаємодіє як з базою знань, так і з базою даних про ліки, щоб зібрати необхідну інформацію. Після завершення аналізу механізм логічного висновку повертає результати до інтерфейсу користувача для представлення користувачеві. Крім того, База даних про ліки постійно доповнює Базу знань найновішою інформацією про ліки, гарантуючи, що прийняття рішень системою ґрунтується на поточних даних.

По суті, експертна система контролю за приймом ліків — це гармонійне поєднання взаємодії з користувачем, обробки даних і управління знаннями. Його конструкція надає перевагу ефективному інформаційному потоку та

точному аналізу даних, що робить його надійним інструментом у сфері управління та контролю над ліками.

2.5 Висновок

Запропоновано математичну модель для контролю прийняття ліків, за допомогою якої обчислюються зведений показник добробуту, зміна стану та оцінка ризику, що допомагає точніше конролювати прийом ліків.

Розроблено загальну структурну схему інформаційної технології контролю прийняття ліків, в основі якої покладена математична модель композитного показинка добробуту, що дозволяє підвищити точність надання рекомендації щодо контролю прийняття ліків. Система складається з модулів: клієнтська частина (відповідає за відображення даних та взаємодію з користувачем), експертна система (аналізує стан) а також серверна частина (взаємодіє з базою даних та експертною системою).

Наведено діаграму діяльності інформаційної технології контролю прийняття ліків.

Розроблено структурну схему системи контролю прийняття ліків та наведено роль кожного рівня системи.

Розроблено алгоритм роботи системи контролю прийняття ліків, який передбачає використання експертної системи, що дозволяє підвищити точність контролю прийняття ліків.

Обґрунтовано доцільність вибору системи управління для бази даних, а саме MsSql, яка дозволяє швидко взаємодіяти з великими об’ємами даних та містить переваги реляційної СУБД. Здійснено нормалізацію бази даних до третьої нормальної форми.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ КОНТРОЛЮ ПРИЙНЯТТЯ ЛІКІВ

3.1 Обґрунтування вибору програмних засобів та мови програмування для реалізації мобільного застосунку

Для розробки програмного забезпечення для адміністрування ліків необхідно використовувати мову програмування. До них належать C#, Java та JavaScript. Розглянемо характеристики кожного з них докладніше.

Java — це об’єктно-орієнтована мова програмування, яка в основному використовується для написання веб-додатків і програм для Android. Програми Java компілюються в байт-код, який потім виконується JVM (Віртуальна машина Java). JVM — це програма, яка обробляє байт-код і надсилає інструкції на машину як інтерпретатор. Однією з особливостей мови програмування Java є гнучка система безпеки, оскільки виконання програми повністю контролюється віртуальною машиною JVM, тому будь-яка операція, яка перевищує вказану потужність машини, викликає негайне переривання. До переваг мови програмування Java можна віднести високу безпеку виконання коду, численні готові рішення від бібліотек до фреймворків і велику спільноту, яка формувалася десятиліттями [12].

C# — безпечна для типів об’єктно-орієнтована мова програмування для платформи .NET. Це проста, але потужна мова програмування, яка дозволяє програмістам створювати багатоцільові програми.

Синтаксис мови програмування C# близький до Java. Мова має сувору статичну типізацію, підтримує поліморфізм, перевантаження операторів, покажчики на функції-члени класу, властивості, події, властивості, винятки, коментарі у форматі XML [13].

Програми на C# працюють у .NET, віртуальній системі виконання, яка викликає загальномовне середовище виконання (CLR) і бібліотеки класів. CLR є реалізацією Common Language Infrastructure (CLI), міжнародного стандарту

від Microsoft. CLI — це основа для створення середовищ виконання та розробки, де мови та бібліотеки взаємодіють прозоро.JavaScript – динамічна, об'єктно-орієнтована прототипна мова програмування. Найчастіше використовується для створення веб-сайтів та мобільних додатків, що надає можливість на боці клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд.

JavaScript класифікується як прототипна (підмножина об’єктно-орієнтованої мови) і є динамічно типізованою мовою сценаріїв. На додаток до створення прототипів, JavaScript підтримує інші парадигми програмування (імперативні та частково функціональні) і пов’язані архітектурні функції, такі як динамічна та слабка типізація, автоматичне керування пам’яттю, успадкування прототипів і функціонує як об’єкти першого класу. Підтримується частково [14].

Враховуючи призначення та функціональність, необхідну для роботи фармацевтичного додатку, можна зробити висновок, що використання мови програмування JavaScript є більш доцільним, оскільки вона в основному використовується для створення гібридних мобільних додатків.

Вибір каркаса також важливий. На даний момент існує три основні структури для розробки мобільних додатків на мові програмування JavaScript. До них належать React Native, Ionic і NativeScript. Кожен вибір залежить від багатьох факторів, включаючи завдання, бюджет і тип проекту.

Насправді React Native вважається найкращою бібліотекою JavaScript для створення нативних програм для всіх пристроїв і платформ. За допомогою React Native ви можете розробляти різноманітні програми для iOS та Android. Він також надає можливість створювати версії різних компонентів для певної платформи, що полегшує використання єдиної кодової бази на різних платформах. Мобільні програми, такі як Facebook, Instagram, Skype і Discord, написані з використанням RN [15].

Native Script — це платформа з відкритим вихідним кодом для створення нативних мобільних додатків, яка підтримує Angular, TypeScript, JavaScript, CSS і Vue [16].

Ionic допомагає створювати гібридні та прогресивні веб-програми, а також кросплатформні програми. Ця платформа з відкритим кодом пропонує преміальні послуги зі створення додатків. Ionic включає створення веб-додатків, додатків для Android та iOS [17].

React Native є чудовим варіантом для розробки мобільних додатків для наркотиків через такі причини:

– Чудовий набір інтерфейсу користувача для прискорення процесу створення красивих і корисних інтерфейсів користувача.
– В усіх відношеннях RN значно перевершує своїх конкурентів за продуктивністю, дозволяючи користувачам швидше бачити потрібну інформацію.
– Просте підключення сторонніх плагінів і API, включаючи картки та платіжні системи.
– Час, необхідний для написання та підтримки коду в React Native, дуже малий, що дозволяє швидко розробляти невеликі програми.
– Можливість розробки гібридних додатків (для iOS та Android). Це дозволяє використовувати одну кодову базу.

Результатом є каркас, який ідеально відповідає нашому завданню. Створюйте мобільні медичні програми швидко та з мінімальними ресурсами. Є можливість підключення сторонніх плагінів (відкрита аптечна картка) і використання вбудованого UI kit для створення приємного інтерфейсу для користувачів. Оскільки React Native є нативним, є можливість додати push-повідомлення, щоб нагадувати користувачам прийняти ліки.

3.2 Обгрунтування вибору типу архітектури застосунку

Під час реалізації застосунків з великою кількістю функціоналу, таких як додаток для контролю прийняття ліків з часом постає проблема масштабування проекту. Зі збільшенням кількості модулів та функцій стрімко зростає кількість витрачених ресурсів для розробки та підтримки проекту. На прикладі застосунку для контролю прийняття ліків, який має великий набір функціоналу, такий як: персональний кабінет, історія прийняття ліків, модуль нагадування про прийом ліків, календар та інші, розглянемо монолітну та мікросервісну архітектуру.

Монолітна архітектура — це традиційна модель програмного забезпечення, яка побудована як уніфікована одиниця, самодостатня та незалежна від інших програм. Ця архітектура представляє собою окрему велику обчислювальну мережау з єдиною кодовою базою, яка об'єднує всі бізнес-завдання. Слово «моноліт» часто приписують чомусь великому та льодовиковому, що не далеко від істини монолітної архітектури для розробки програмного забезпечення [18]. Для того щоб внести зміни в програму такого типу, потрібно оновити весь стек, отримавши доступ до бази коду та створивши та розгорнувши оновлену версію інтерфейсу служби. Це робить оновлення обмежувальними та забирає багато часу розробника [18].

Мікросервісна архітектура — це архітектурний стиль для побудови програмних систем шляхом розбиття складних програм на набір невеликих, слабко пов'язаних і незалежно розгорнутих сервісів [19]. Кожен сервіс в архітектурі мікросервісів відповідає за певний, чітко визначений набір функцій і може спілкуватися з іншими сервісами через мережу.

На сьогоднішній день мікросервісна архітектура масово використовується для розробки серверного програмного забезпечення, проте майже не використовується для клієнтського, через свою новизну саме для клієнтських застосунків та складність в налаштуванні процесів. Проте даний

підхід має значно більше переваг для великих проектів порівняно з монолітною архітектурою.

Переваги мікросервісної архітектури (мікрофронтенди) відносно монолітної:

1. Масштабування: ми можемо масштабувати окремі мікроіфронтенди в залежності від їхніх конкретних вимог.

2. Незалежна розробка: різні команди та розробники можуть працювати над окремими мікрофронтендами, не впливаючи один на одного, що сприяє паралельній розробці та пришвидшить кінцевий результат.

3. Модульність: мікрофронтенди дозволяють розділяти програмне забезпечення на менші самостійні модулі, що полегшує керування та незалежну розробку окремих функцій або компонентів.

4. Гнучкість: ми можемо використовувати різні технології, фреймворки або бібліотеки в окремих мікрофронтендах без необхідності робити зміни в інших сервісах.

5. Ізоляція: ізоляція між мікрофронтендами зменшує ризик помилок в одному модулі, що також призводить до того, що помилка в одному модулі не виведе з ладу всю програму.

Проте є й певні недоліки мікрофронтендів:

1. Складність: мікрофронтенди створюють додаткову складність з точки зору маршрутизації, зв’язку та координації між різними компонентами.

2. Проблеми з комунікацією: зв’язок між мікрофронтендами може бути складним і потребуватиме спеціальних рішень або бібліотек для забезпечення плавної інтеграції та обміну даними.

3. Контроль версій і сумісностей: керування версіями та забезпечення сумісності всіх мікрофронтендів між собою може бути складним завданням, що може призвести до потенційних проблем з інтеграцією.

Для того, щоб реалізувати мікросервісну архітектуру для клієнтського програмного забезпечення зазвичай використовують ModuleFederationPlugin [20]. ModuleFederationPlugin дозволяє збірці надавати або споживати модулі з

іншими незалежними збірками під час виконання, проте можна самостійно налаштувати процес збірки.

Поради щодо реалізації мікросервісної архітектури для React застосунків:

- Потрібно створити Core Application який буде відповідати за підвантаження збірок ваших сервісів, вмонтовування та розмонтовування скриптів у DOM — Document Object Model [21].
- Кожен дочірній сервіс, що буде використовуватись в корі повинен мати однакову структуру збірки та функціонал, що дозволить глобально викликати (з об’єкта window) методи для монтування та розмонтування даного сервісу.
- Важливо, щоб в кор сервіса був доступ до збірок дочірніх сервісів, тому потрібно розміщувати сервіси в межах Docker контейнера, або ж ігнорувати CORS — Cross-Origin Resource Sharing [22] (не рекомендується через можливі вразливості сервісів).
- Використання окремої бібліотеки з набором повторюваних компонентів та функцій допоможе уникнути дублювання коду в сервісах.
- Для комунікації між сервісами можна використовувати всі доступні методи браузера такі як: localStorage, sessionStorage, CustomEvent, window, serviceWorkers та інші.

Аналізуючи всі переваги та недоліки можна дійти висновку, що за наявності ресурсів мікросервісна архітектура краще підходить для розробки великих клієнтський застосунків, таких як застосунок для контролю прийняття ліків, через свою гнучкість та масштабування. Адже над кожним модулем можуть незалежно працювати різні команди та розробники і при зміні в певному модулі не потрібно збирати ввесь проект, а лише даний модуль. Також через можливість розділення великих модулів, таких як календар та модуль історії прийняття ліків, функціонал стає ізольованим і у випадку помилки в одному з модулів застосунок і далі працюватиме. Через розділення на мікрофронтенди ми маємо можливість використовувати різний набір компонентів для персонального кабінету та календаря.

Схема мікросервсної архітектури зображена на рисунку 3.1.

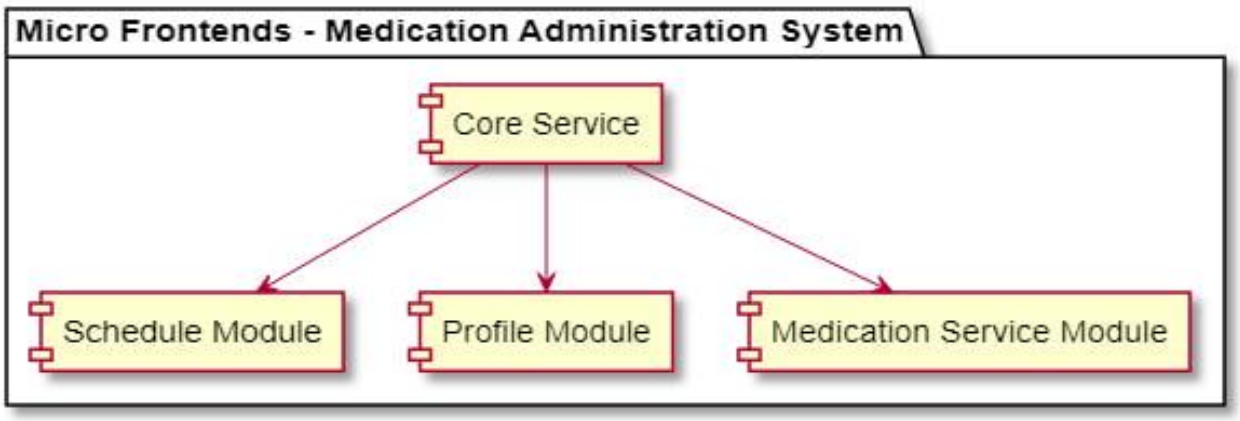

Рисунок 3.1 – Схема мікросервісної архітектури клієнтської частини застосунку

3.3 Обгрунтування вибору технологій та середовища для серверної розробки

При розробці експертної системи для контролю прийняття ліків вирішальним є вибір програмного забезпечення та мов програмування. Цей вибір обумовлений потребою в ефективній обробці даних, потужних обчислювальних можливостях і здатності обробляти великі обсяги даних, часто розподілених між кількома серверами.

Node.js стає основним вибором для розробки бекенда цієї експертної системи, завдяки його ефективній обробці операцій введення/виведення та його придатності для архітектури мікросервісів. Серверна система, побудована на мікросервісній архітектурі, забезпечує гнучкість і масштабованість, необхідні для такої складної програми.

Node.js, середовище виконання JavaScript, побудоване на механізмі JavaScript V8 Chrome, славиться своєю неблокуючою архітектурою, керованою подіями. Завдяки цій якості він надзвичайно добре підходить для додатків, які вимагають обробки даних у режимі реального часу, що є основною вимогою в системах контролю введення ліків [23].

Однопотоковий характер Node.js у поєднанні з його здатністю ефективно обробляти кілька одночасних запитів забезпечує високу продуктивність і масштабованість. Це особливо корисно в управлінні частими та різноманітними запитами даних, типовими для систем контролю ліків.

В екосистемі Node.js Express.js [24] виділяється як мінімальний і гнучкий фреймворк веб-додатків Node.js. Він надає надійний набір функцій для розробки API і веб-додатків.

Express.js спрощує завдання побудови ефективних і масштабованих RESTful API, важливого компонента для архітектури мікросервісів в експертній системі. Це полегшує безперебійний зв’язок між різними мікросервісами та центральною базою даних.

Прийняття архітектури мікросервісів дозволяє розбивати систему на менші служби, які можна розгортати незалежно. Ця модульність покращує ремонтопридатність і сприяє безперервній інтеграції та розгортанню.

Кожен мікросервіс може бути призначений для певної функції, як-от керування даними пацієнтів, планування прийому ліків, відстеження відповідності та сповіщення. Такий поділ проблем призводить до більш керованої та масштабованої розробки системи.

MongoDB, база даних NoSQL, ідеально підходить для керування неструктурованими або напівструктурованими даними. Його гнучкість у обробці різних форматів даних є перевагою для різних типів даних, які зустрічаються в системах контролю ліків [25].

Для структурованих даних перевагу надають базам даних SQL через надійне керування транзакціями та складні можливості запитів. Вони забезпечують надійність, необхідну для обробки структурованих даних пацієнтів і ліків.

Безпека в програмах Node.js зміцнюється за допомогою різноманітних бібліотек і практик, таких як OAuth для безпечної автентифікації та протоколи HTTPS для зашифрованої передачі даних.

Дотримання правил охорони здоров’я, таких як HIPAA, має вирішальне значення для обробки конфіденційних даних пацієнтів. Node.js підтримує різні інструменти та бібліотеки, які відповідають цим суворим стандартам, забезпечуючи безпеку та конфіденційність даних.

Підсумовуючи, вибір Node.js разом із Express.js і архітектурою мікросервісів забезпечує потужну базову структуру для експертної системи в контролі надходження ліків. Ця комбінація пропонує основні можливості, необхідні для системи, включаючи ефективну обробку даних, масштабованість і здатність обробляти різноманітні типи даних безпечно та відповідно до стандартів охорони здоров’я.

Вибір правильного середовища розробки має вирішальне значення для ефективної та результативної розробки системи. Цей вибір залежить від різних факторів, включаючи мовну спеціалізацію, інструменти та загальну ефективність процесу розробки. Після розгляду кількох популярних інтегрованих середовищ розробки (IDE) WebStorm виділяється як найбільш відповідний вибір для потреб нашого проекту, особливо враховуючи його узгодженість із нашим стеком технологій, який включає JavaScript, Node.js і React Native.

WebStorm, розроблений спеціально для JavaScript і пов’язаних із ним технологій, пропонує значні переваги в розумній допомозі з кодом, що має ключове значення для розробки нашого проекту з використанням JavaScript. Ця IDE чудово забезпечує інтегроване середовище, яке підтримує не лише JavaScript, але й пов’язані технології, покращуючи досвід розробки. На відміну від аналогів, WebStorm оснащений комплексним набором вбудованих інструментів для налагодження, тестування та контролю версій, що зменшує залежність від зовнішніх плагінів і спрощує робочий процес.

Одним із вирішальних факторів є ергономічний дизайн і інтелектуальні функції WebStorm, які створені для підвищення продуктивності розробників. Ці функції дозволяють ефективніше кодувати, швидше виявляти помилки та ефективний рефакторинг, що є важливим у проекті такого масштабу та

складності. Хоча WebStorm є комерційним продуктом, для якого потрібна платна ліцензія, інвестиції виправдані значною економією часу та підвищеною ефективністю, яку він пропонує.

Для порівняння, інші IDE, як-от Visual Studio Code, Atom і Sublime Text, хоч і потужні самі по собі, вимагають додаткового налаштування та налаштування, щоб досягти того самого рівня функціональності для нашого конкретного випадку використання. Крім того, фокус WebStorm на веб-розробці та розробці мобільних пристроїв ідеально узгоджується з напрямком нашого проекту, що робить його найбільш підходящим середовищем для розробки системи контролю за адмініструванням ліків.

Підсумовуючи, рішення про використання WebStorm як середовища розробки є стратегічним вибором, який відповідає нашим технологічним потребам і розширює наші можливості розробки. Його спеціалізована увага на JavaScript, комплексні інструменти розробки та функції, призначені для підвищення продуктивності, роблять його безцінним активом для успішного розвитку нашої системи.

3.4 Програмна реалізація інформаційної технології контролю прийняття ліків

Виходячи з аналізу предметної області було виділено наступні сутності:

1. User – відповідає за збереження інформації про користувача: ім’я, логін, пароль, аватар, вік, зріст, віга, стать та хвороби. Програмну реалізацію наведено на рисунку 3.2.

```typescript
@Sceema()
export class User {
  @Prop()
  public name: string;

  @Prop()
  public email: string;

  @Prop()
  private password: string;

  @Prop()
  public avatar: string;

  @Prop({ required: true })
  public age: number;

  @Prop({ required: true })
  public height: number;

  @Prop({ required: true })
  public weight: number;

  @Prop({ required: true })
  public gender: string;

  @Prop({ default: [] })
  public diseases: string[];

  @Prop({ default: [] })
  public treatment: UserTreatment[]
}
```

Рисунок 3.2 – Програмна реалізація сутності User

2. UserTreatment – відповідає за збереження інформації про ліки, які вживає користувач, а саме: назва препарату, тип препарату, час прийому та кількість прийому ліків. Програмну реалізацію наведено на рисунку 3.3.

```
@Sceema()
export class UserTreatment {
  @Prop()
  public title: string;

  @Prop()
  public unit: string;

  @Prop()
  private frequency: number;

  @Prop()
  public intake: string;
}
```

Рисунок 3.3 – Програмна реалізація сутності UserTreatment

3. Medicine – сутність, що містить інформацію про ліки, а саме: назва, опис, протипоказання. Програмну реалізацію наведено на рисунку 3.4.

```
@Sceema()
export class Medicine {
  @Prop()
  public title: string;

  @Prop()
  public description: string;

  @Prop()
  private contraindication: string;
}
```

Рисунок 3.4 – Програмна реалізація сутності Medicine

Розроблені сутності дозволяють зручно працювати з предметною областю та виконують всі поставленні функції. Також, при модифікації додатку та з розширенням функціоналу, їх буде легко масштабувати та доповнити необхідним функціоналом.

Для реалізації клієнтської частини додатку для прийому ліків було спроектовано та реалізовано наступні модулі:

1. AuthScreen – компонент, який відповідає за реєстрацію та авторизацію користувача.
2. SearchScreen – компонент, який відповідає з пошук інформації про певні медичні препарати.
3. HistoryScreen – компонент, який відповідає за відображення інформації та графіків про прийом ліків.
4. TreatmentScreen – компонент, який відповідає за відображення добавлених користувачем препаратів.
5. CreateTreatmentScreen – модуль, який відповідає за добавлення користувачем відповідного препарату.
6. ProfileScreen – компонент, який відповідає за перегляд та редагування інформації про користувача.
7. NotificationService – модуль, який відповідає за відправку нагадувань про прийом ліків.

Реалізація системи контролю прийняття ліків із використанням мікросервісної архітектури представляє сучасний підхід до створення масштабованих, ефективних і надійних програмних додатків. Використовуючи потужність React Native для інтерфейсу та Node.js для мікросервісів бекенда, система може запропонувати бездоганну взаємодію з користувачем, зберігаючи високу продуктивність і гнучкість.

1. Розробка інтерфейсу за допомогою React Native

React Native, популярний фреймворк для створення мобільних додатків, обрано для розробки інтерфейсу системи. Це дає змогу створювати кросплатформену мобільну програму, яка забезпечує нативний досвід як для користувачів iOS, так і для Android.

Інтерфейс користувача: програма React Native матиме інтуїтивно зрозумілий інтерфейс користувача, який дозволяє пацієнтам і постачальникам медичних послуг взаємодіяти з системою. Це включає в себе функції для

перегляду графіків прийому ліків, підтвердження прийому ліків і доступу до звітів про відповідність.

Зв’язок із бекендом: інтерфейс зв’язуватиметься з різними мікросервісами бекенда через RESTful API або потенційно за допомогою GraphQL для складніших запитів.

2. Розробка бекенда з Node.js і мікросервісами

Сервер системи побудовано з використанням Node.js, відомого своєю ефективністю та масштабованістю, зокрема в обробці численних операцій введення-виведення. Архітектура мікросервісу поділяє серверну частину на невеликі служби, які можна розгортати незалежно, кожна з яких відповідає за певні функції.

Архітектура мікросервісу: кожен мікросервіс обробляє різні аспекти системи, такі як керування пацієнтами, планування ліків, відстеження відповідності та сповіщення. Цей модульний підхід підвищує зручність обслуговування, масштабованість і легкість розгортання.

Зв’язок між службами: мікросервіси спілкуються один з одним за допомогою спрощених протоколів, таких як HTTP/REST, або черги обміну повідомленнями для асинхронного зв’язку, забезпечуючи ефективний обмін даними.

3. Основні серверні мікросервіси

Служба керування пацієнтами: керує профілями пацієнтів, включаючи особисту інформацію та історію хвороби.

Служба керування ліками: обробляє дані про ліки, зокрема відомості про дозування та можливі взаємодії.

Служба керування графіками: відповідає за створення та керування графіками прийому ліків для кожного пацієнта.

Служба відстеження відповідностей: відстежує та записує дотримання пацієнтами графіків прийому ліків.

Служба сповіщень: надсилає нагадування та сповіщення пацієнтам щодо прийому ліків.

4. Зберігання та керування даними

Системи баз даних: залежно від вимог можна використовувати комбінацію баз даних SQL і NoSQL. Бази даних SQL для структурованих даних, таких як інформація про пацієнтів, і NoSQL для більш гнучких вимог до даних, таких як журнали відповідності.

Безпека даних: застосовуйте шифрування та протоколи безпечного доступу для захисту конфіденційних даних пацієнтів.

5. Інтеграція та тестування

Шлюз API: реалізуйте шлюз API як єдину точку входу для інтерфейсу для взаємодії з різними мікросервісами, спрощуючи архітектуру інтерфейсу та забезпечуючи додатковий рівень безпеки.

Тестування: Ретельне тестування кожного мікросервісу є важливим, включаючи модульне тестування, інтеграційне тестування та наскрізне тестування всієї системи.

6. Розгортання та моніторинг

Контейнерізація: використовуйте такі інструменти контейнеризації, як Docker, для розгортання мікросервісів, забезпечуючи узгодженість у різних середовищах.

Моніторинг і технічне обслуговування: впроваджуйте інструменти моніторингу для відстеження продуктивності та працездатності системи, а також налаштуйте конвеєри безперервної інтеграції/безперервного розгортання (CI/CD) для поточного обслуговування та оновлень.

Програмна реалізація системи контролю адміністрування ліків за допомогою React Native і архітектури мікросервісу на основі Node.js пропонує надійне, масштабоване та ефективне рішення. Ця архітектура не тільки полегшує керування та оновлення окремих компонентів, але й забезпечує високопродуктивну та зручну програму, що має вирішальне значення для чутливої сфери управління ліками.

3.5 Тестування та аналіз результатів роботи інформаційної технології контролю прийняття ліків.

Модульне тестування є критично важливим компонентом у розробці системи контролю за введенням ліків. Ця фаза тестування зосереджена на перевірці найдрібніших частин програми, відомих як одиниці, які зазвичай містять окремі функції або методи в коді. Основна мета полягає в тому, щоб кожен блок працював за призначенням ізольовано, тим самим гарантуючи надійність і якість усієї системи.

Для unit тестування була використана бібліотека react-testing-library за допомогою якої було створено 115 наборів тестів для перевірки компонетів. Результати тестування системи на ведені на рисунку 3.5.

```
Test Suites: 1 skipped, 114 passed, 114 of 115 total
Tests:       2 skipped, 326 passed, 328 total
Snapshots:   8 passed, 8 total
Time:        75.808 s, estimated 86 s
Ran all test suites.
```

Рисунок 3.5 – Результат unit тестування системи

При тестування системи для контролю прийому ліків потрібно було перевірити коректну роботу всіх модулів. Також, виявити та виправити помилки та неточності у роботі програми. Для того, щоб точно впевнитись у коректній роботі закладеного функціоналу та загалом правильності роботи застосунку ми використали e2e тестування, яке допомагає забезпечити правильну поведінку всіх інтерфейсів.

1. Вхід до додатку з правильними даними для входу.

   Вихідні дані: коректний логін та пароль;

Очікуваний результат: успішний вхід до додатку;

Результат, рисунок 3.6:

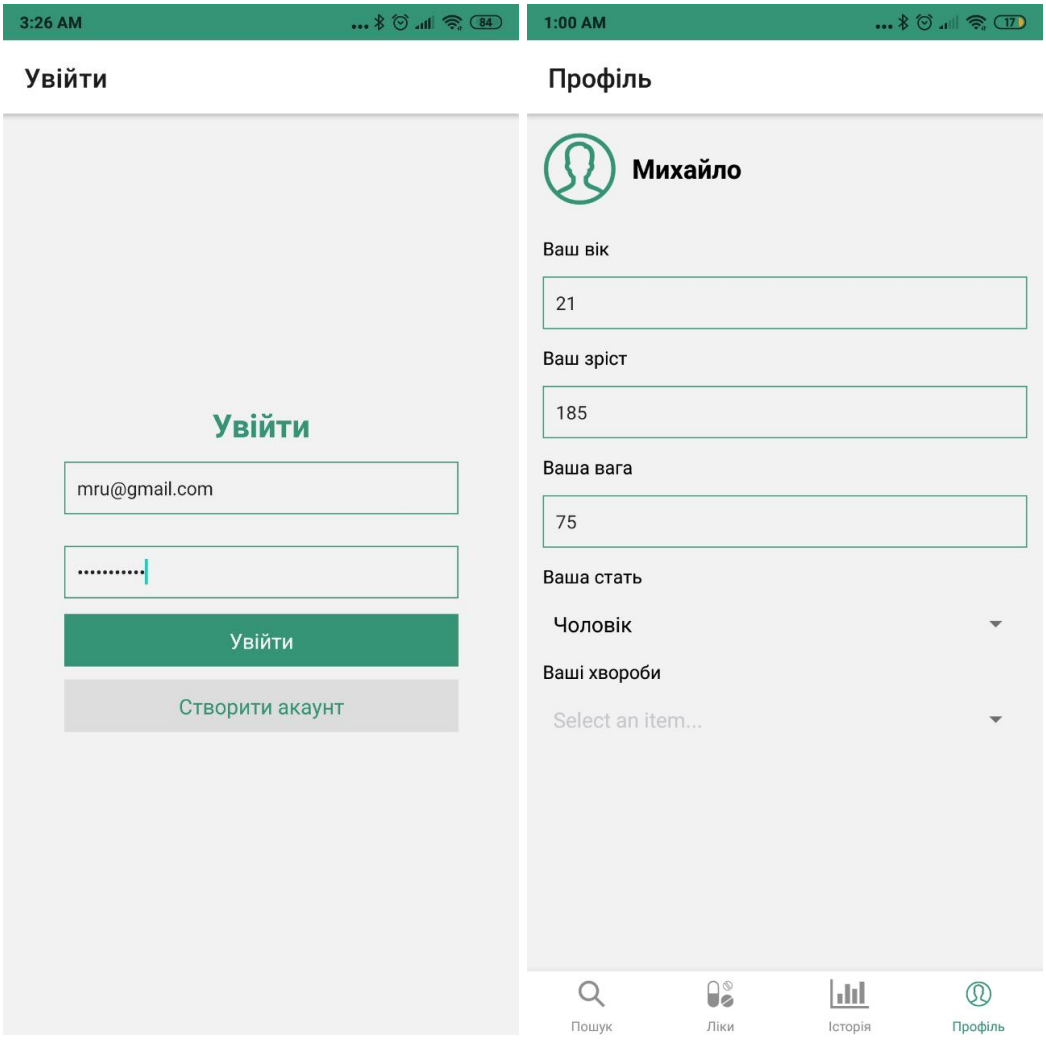

Рисунок 3.6 – Результат тестування входу до додатку з правильними даними

2. Вхід до додатку з неправильними даними для входу.

   Вихідні дані: хибні логін та пароль;

   Очікуваний результат: відсутність доступу;

   Результат, рисунок 3.7:

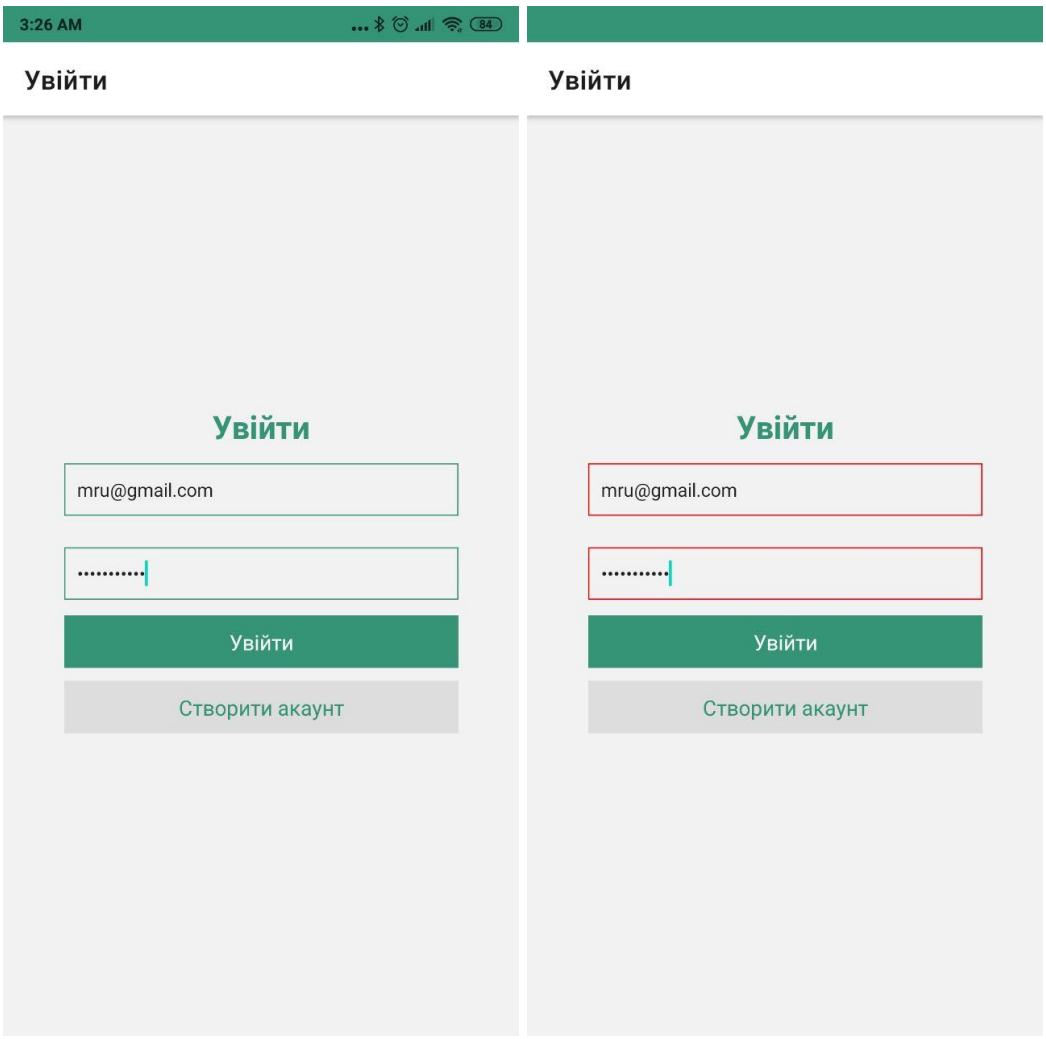

Рисунок 3.7 – Результат тестування входу до додатку з неправильними даними

3. Додавання нового лікарського запису.

   Вихідні дані: Назва препарату – Спазмалгон, Тип ліків – Таблетки, Частота прийому – Один раз на день, Час прийому – 8:00;

   Очікуваний результат: відображення запису після додавання;

   Результат, рисунок 3.8:

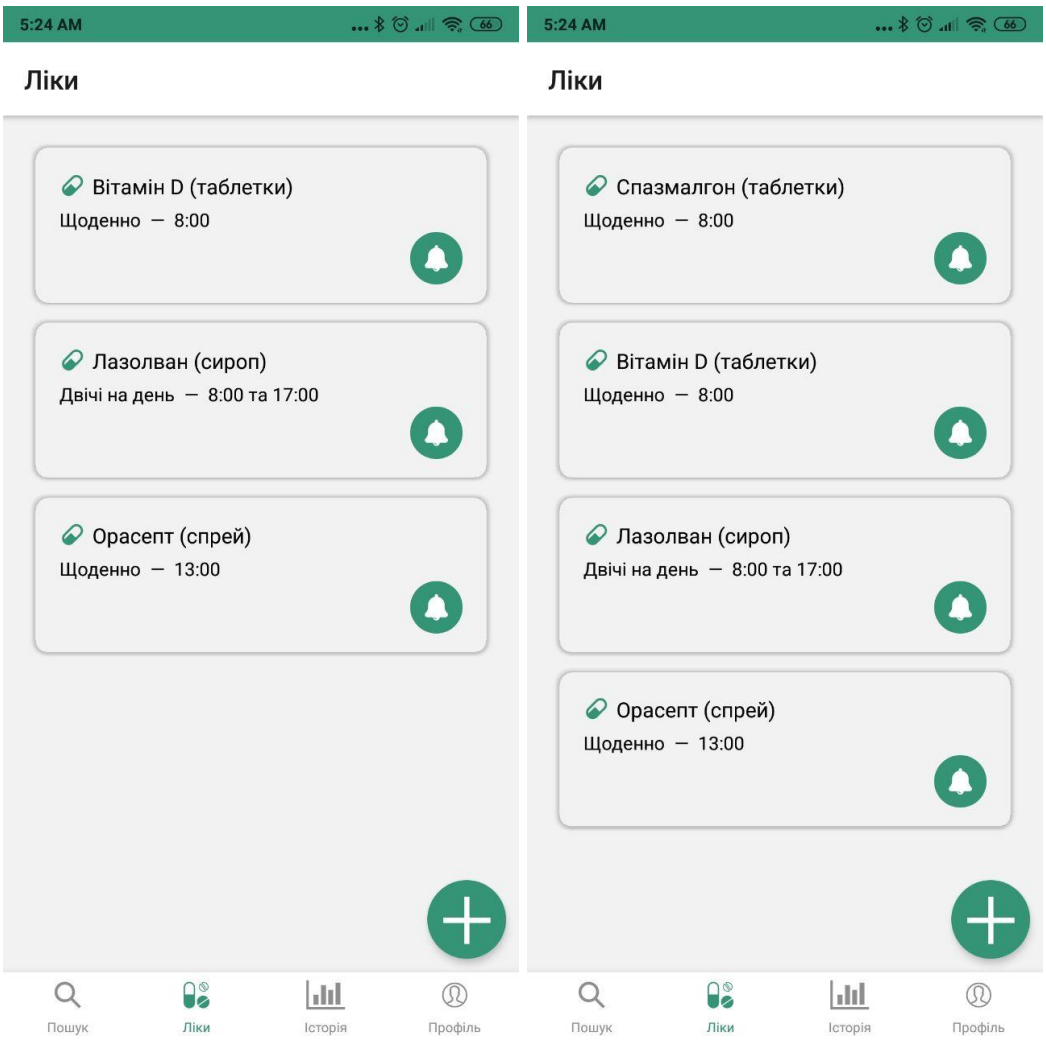

Рисунок 3.8 – Результат тестування додавання нового лікарського засобу

4. Аналіз рекомендацій по прийому лікарського засобу.

   Вихідні дані: Назва препарату, хвороби користувача;

   Очікуваний результат: застереження щодо прийому лікарського засобу;

   Результат, рисунок 3.9:

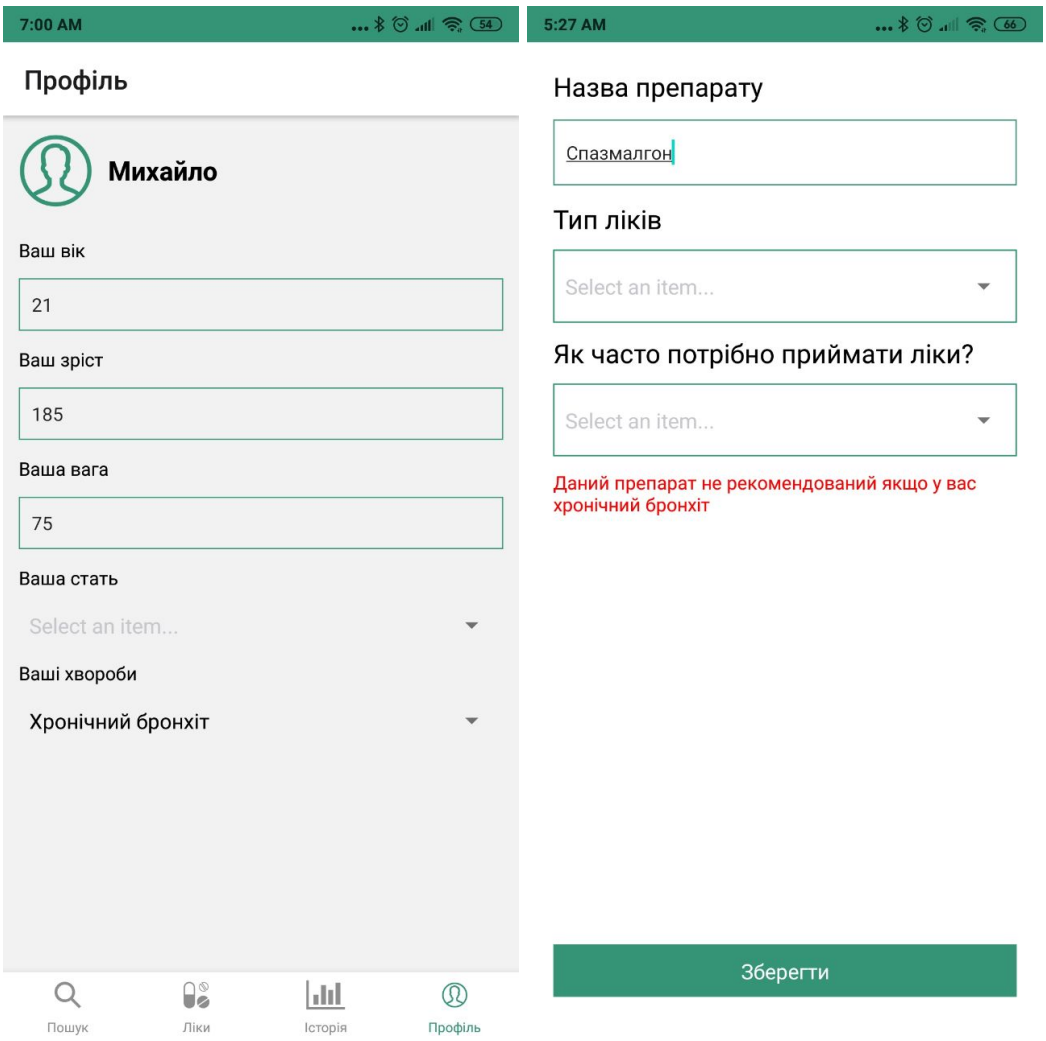

Рисунок 3.9 – Результат тестування аналізу рекомендацій по прийому лікарського засобу

5. Пошук лікарського засобу.

   Вихідні дані: Назва лікарського засобу;

   Очікуваний результат: відображення потрібного лікарського засобу;

   Результат, рисунок 3.10:

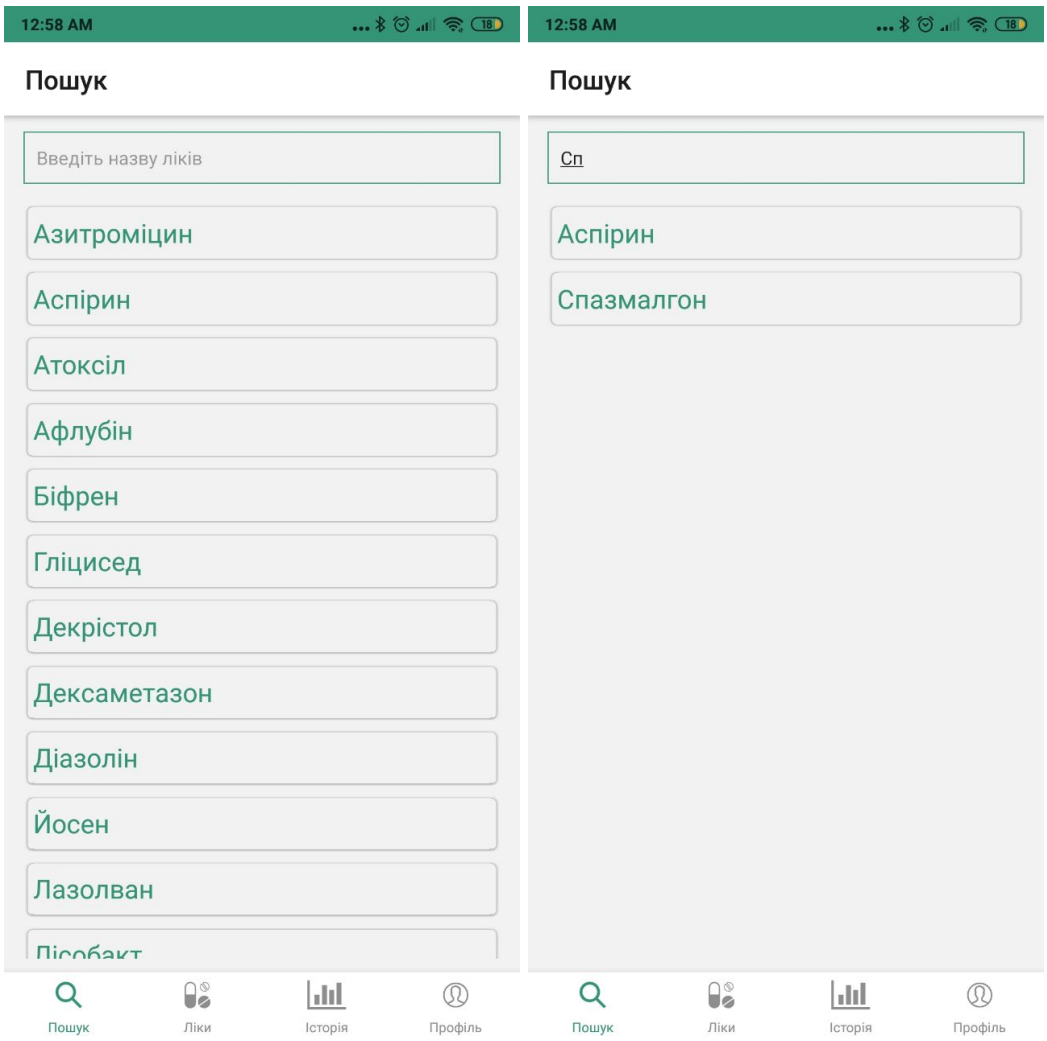

Рисунок 3.10 – Результат тестування пошуку лікарського засобу

6. Зміна інформації про користувача

   Вихідні дані: Вік – 21, Зріст – 185, Вага – 75, Стать – Чоловік, Хвороби – Хронічний бронхіт;

   Очікуваний результат: відображення інформації на сторінці користувача, рисунок 3.11:

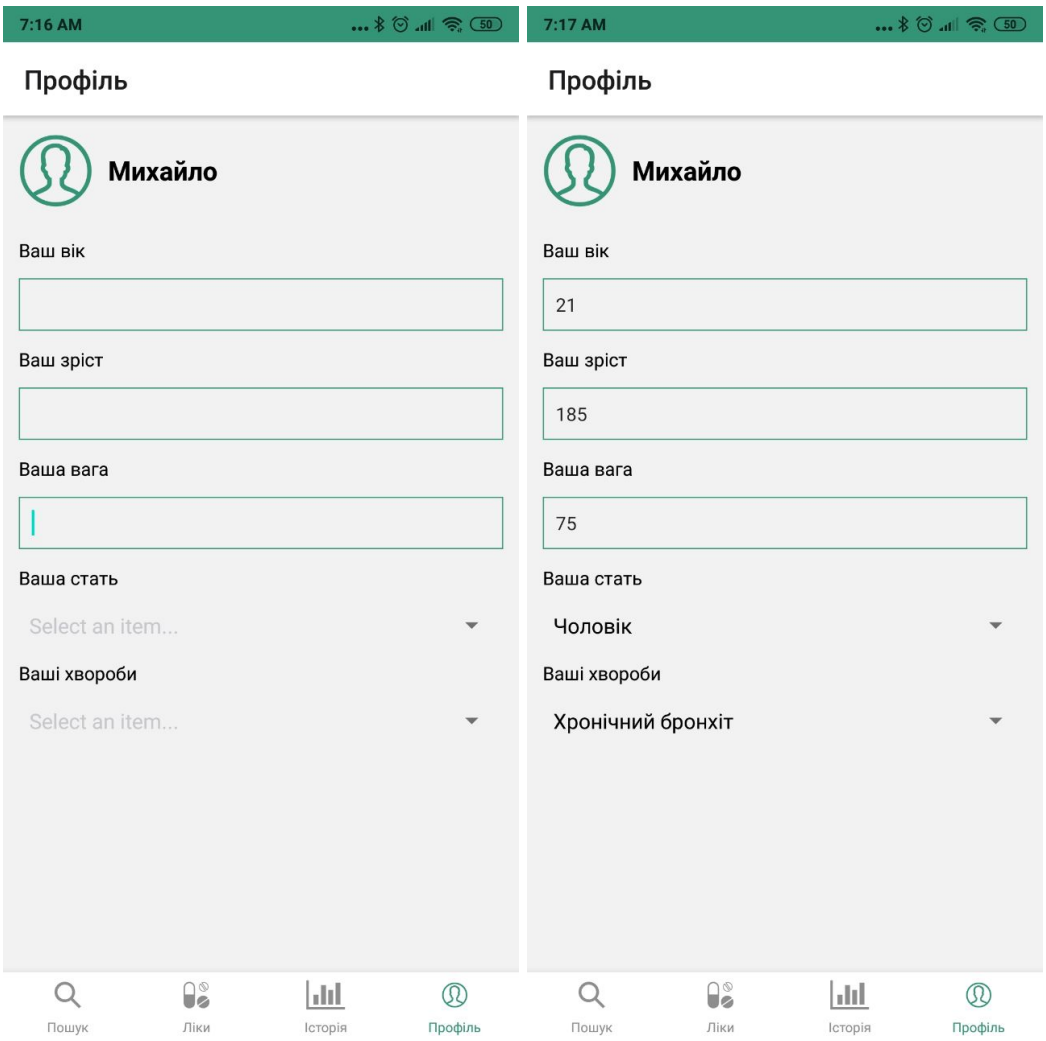

Рисунок 3.11 – Результат тестування зміни інформації про користувача

Розроблений додаток має функціонал, наведений в таблиці 3.1, який було порівняно із функціоналом найпопулярніших мобільних застосунків для прийому ліків, таких як MediSafe App, Dosecast, CareClinic.

Таблиця 3.1 – Порівняльна характеристика розробленого ПЗ

Назва програми	Пошук інформації про препарати	Особистий кабінет	Контроль прийому ліків	Нагадування про прийом ліків
MediSafe App	+/-	+	-	+/-
Dosecast	+	-	-	+
CareClinic	-	+	-	+
Розроблений додаток	+	+	+	+

Отже, розроблений додаток для прийому ліків було протестовано на відповідність меті дослідження. Функціонал програмного забезпечення є розширеним по відношенню до існуючих сучасних рішень та відповідає завданню.

3.6 Висновок

Обґрунтовано вибір мови програмування JavaScript для розробки інформаційної технології контролю прийняття ліків, а також фреймворк React Native, через його крос-платформенні можливості та бездоганну інтеграцією з Node.js, що підтверджує його придатність для розробки зручного та адаптивного застосунку. Цей вибір гарантує, що програма ефективно задовольняє різноманітні потреби як пацієнтів, так і постачальників медичних послуг.

Здійснено вибір мікросервісної архітектури для серверної частини на базі Node.js підкреслює рух до модульної, масштабованої та зручної для обслуговування системи. Ця архітектура сприяє незалежному масштабуванню сервісів і підвищує стійкість системи, що має вирішальне значення для управління складними завданнями.

Розроблено інформаційну технологію контролю прийняття ліків із застосуванням новітніх технологій таких як Express та stugna-es. Відокремлено клієнтську частину та серверну, що дає змогу застосовувати серверну частину в інших системах, незалежно від клієнтської, використовуючи API запити.

Здійснено тестування розробленого програмного забеспечення, яке показало правильність виконання поставлених задач та відповідність поставленим технічним вимогам.

4 ЕКОНОМІЧНА ЧАСТИНА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ КОНТРОЛЮ ПРИЙНЯТТЯ ЛІКІВ

4.1 Проведення комерційного та технологічного аудиту інформаційної технології контролю прийняття ліків.

Метою проведення комерційного і технологічного аудиту є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності, тобто під час виконання магістерської кваліфікаційної роботи.

Для проведення комерційного та технологічного аудиту залучаємо 3-х незалежних експертів, якими є провідні викладачі випускової або спорідненої кафедри.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу здійснюємо із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями, а результати зводимо до таблиці 4.1.

Таблиця 4.1 – Результати оцінювання науково-технічного рівня і комерційного потенціалу засобу поляриметричного аналізу оптично активних рідни

Критерії	Експерти		
	Експерт 1	Експерт 2	Експерт 3
	Бали, виставлені експертами		
Технічна здійсненність концепції	2	2	2
Ринкові переваги (наявність аналогів)	3	1	3
Ринкові переваги (ціна продукту)	2	2	3
Ринкові переваги (технічні властивості)	3	1	2
Ринкові переваги (експлуатаційні витрати)	3	2	3
Ринкові перспективи (розмір ринку)	2	2	2
Ринкові перспективи (конкуренція)	3	2	3

Продовження таблиці 4.1

Практична здійсненність (наявність фахівців)	2	2	2
Практична здійсненність (наявність фінансів)	3	2	1
Практична здійсненність (необхідність нових матеріалів)	3	3	3
Практична здійсненність (термін реалізації)	2	2	3
Практична здійсненність (розробка документів)	3	3	2
Сума балів	31	24	29
Середньоарифметична сума балів, СБ	28		

За результатами розрахунків, наведених в таблиці 1 робимо висновок про те, що науково-технічний рівень та комерційний потенціал інформаційної технології контролю прийняття ліків – середній.

4.2Розрахунок витрат на здійснення науково-дослідної роботи

Витрати на оплату праці. Належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці, також будь-які види грошових і матеріальних доплат, які належать до елемента «Витрати на оплату праці».

Основна заробітна плата дослідників. Витрати на основну заробітну плату дослідників ($З_о$) розраховують відповідно до посадових окладів працівників, за формулою 4.1:

$$З_о = \sum_{i=1}^{к} \frac{М_{ni} \cdot t_i}{Т_р}, \quad (4.1)$$

де к – кількість посад дослідників, залучених до процесу дослідження; $М_{ni}$ – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.; $Т_р$ – число робочих днів в місяці; приблизно $Т_р$ = (21…23) дні; t_i – число робочих днів роботи розробника (дослідника).

Зроблені розрахунки зводимо до таблиці 4.2.

Таблиця 4.2 – Витрати на заробітну плату дослідників

Посада	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник	13500	643	21	13500
Розробник	12500	595	21	12500
Консультанти	12500	595	10	5950
Всього:	31950			

Основна заробітна плата робітників. Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт розраховують за формулою 4.2:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i, \quad (4.2)$$

де $С_і$ – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год; t_i – час роботи робітника на виконання певної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду С і можна визначити за формулою 4.3:

$$С_і = \frac{М_м \cdot К_і \cdot К_с}{Т_р \cdot t_{зм}}, \quad (4.3)$$

де $М_м$ – розмір прожиткового мінімуму працездатної особи або мінімальної місячної заробітної плати (залежно від діючого законодавства), у 2023 році $М_м$=6700 грн; $К_і$ – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду; $К_с$ – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати; $Т_р$ – середня кількість робочих днів в місяці, приблизно $Т_р$ = 21…23 дні; $t_{зм}$ – тривалість зміни, год.

Таблиця 4.3 – Витрати на заробітну плату робітників

Найменування робіт	Трудомісткість, н-год.	Розряд роботи	Погодинна тарифна ставка	Тариф. коеф.	Величина, грн.
Установка електронно-обчислювального обладнання	6	2	43,5	1,09	261
Підготовка робочого місця дослідника	2	2	43,5	1,09	87
Налагодження програмних блоків	6	5	54,2	1,36	325
Тестування системи	8	2	43,5	1,09	348
Всього					1021

Додаткова заробітна плата. Додаткова заробітна плата $З_д$ всіх розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховується як (10…12)% від суми основної заробітної плати всіх розробників та робітників, тобто:

$$З_д = 0{,}1 \cdot (З_о + З_р) = 0{,}1 \cdot (31950 + 1021) = 3297 \text{ грн.}$$

Відрахування на соціальні заходи. Нарахування на заробітну плату $Н_{зп}$ розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою:

$$Н_{зп} = \beta \cdot (З_о + З_р + З_д) =$$
$$= 0{,}22 \cdot (31950 + 1021 + 3297) = 7979 \text{ грн.}$$

де $З_о$ – основна заробітна плата розробників, грн.; $З_р$ – основна заробітна плата робітників, грн.; $З_д$ – додаткова заробітна плата всіх розробників та робітників, грн.; β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, % (приймаємо для 1-го класу професійності ризику 22%).

Спецустаткування для наукових (експериментальних) робіт. Вартість спецустаткування визначається за прейскурантом гуртових цін або за даними базових підприємств за відпускними і договірними цінами 4.4:

$$В_{спец} = \sum_1^к Ц_і \cdot С_{пр.і} \cdot К_і, \quad (4.4)$$

де $Ц_і$ – ціна придбання спецустаткування і-го виду, грн.; $С_{пр.і}$ – кількість одиниць спецустаткування відповідного виду, шт.; $К_і$ – коефіцієнт транспортних витрат, $К_і$ = (1,1…1,15); n – кількість видів спецустаткування.

Таблиця 4.4 – Витрати на придбання спецустаткування

Найменування спецустаткування	Ціна за одиницю, грн.	Витрачено	Вартість спецустаткування, грн.
Засоби передачі даних	11650,00	1	11650
Всього, з врахуванням коефіцієнта транспортних витрат			12815

Програмне забезпечення. До балансової вартості програмного забезпечення входять витрати на його інсталяцію, тому ці витрати беруться

додатково в розмірі 10…12% від вартості програмного забезпечення. Балансову вартість програмного забезпечення розраховують за формулою 4.5:

$$В_{прг} = \sum_{1}^{к} Ц_{іпрг} \cdot С_{прг.і} \cdot К_{і}, \quad (4.5)$$

де $Ц_{іпрг}$ – ціна придбання програмного забезпечення і-го виду, грн.; $С_{прг.і}$ – кількість одиниць програмного забезпечення відповідного виду, шт.; $К_{і}$ – коефіцієнт, що враховує інсталяцію, налагодження програмного забезпечення, $К_{і}$ = (1,1…1,12); к – кількість видів програмного забезпечення.

Таблиця 4.5 – Витрати на придбання програмного забезпечення

Найменування програмного забезпечення	Ціна за одиницю, грн.	Витрачено	Вартість програмного забезпечення, грн.
ОС Windows	8415	1	8415
Прикладний пакет Microsoft Office	7810	1	7810
Програмний засіб IDE – WebStorm	7560	1	7560
Всього, з врахуванням коефіцієнта інсталяції та налагодження			26164

Амортизація обладнання. Амортизація обладнання, комп'ютерів та приміщень, які використовувались під час (чи для) виконання даного етапу роботи.

У спрощеному вигляді амортизаційні відрахування А в цілому бути розраховані за формулою 4.6:

$$А = \frac{Ц_{б}}{Т_{в}} \cdot \frac{t}{12}, \quad (4.6)$$

де $Ц_{б}$ – загальна балансова вартість всього обладнання, комп'ютерів, приміщень тощо, що використовувались для виконання даного етапу роботи, грн.; t – термін використання основного фонду, місяці; $Т_{в}$ – термін корисного використання основного фонду, роки.

Таблиця 4.6 – Амортизаційні відрахування за видами основних фондів

Найменування	Балансова вартість, грн.	Строк корисного використання, років	Термін використання, місяців	Сума амортизації, грн.
Персональний комп'ютер	24370	5	1	406
Робоче місце дослідника	7880	3	1	219
Пристрої виводу інформації	6875	3	1	191
Всього	816			

Витрати на електроенергію для науково-виробничих цілей. Витрати на силову електроенергію $В_е$, якщо ця стаття має суттєве значення для виконання даного етапу роботи, розраховуються за формулою:

Таблиця 4.7 – Витрати на електроенергію

Найменування обладнання	Потужність, кВт	Тривалість годин роботи
Персональний комп'ютер	0,45	160
Робоче місце дослідника	0,15	160
Пристрої виводу інформації	0,03	160
Засоби передачі даних	0,05	120

$$Ве = \sum \frac{W_i \cdot t_i \cdot Ц_е \cdot К_{впі}}{ККД}$$

$$= \frac{0,45 \cdot 160 \cdot 7,5 \cdot 0,95}{0,98} + \frac{0,15 \cdot 160 \cdot 7,5 \cdot 0,95}{0,98}$$

$$+ \frac{0,03 \cdot 160 \cdot 7,5 \cdot 0,95}{0,98} + \frac{0,05 \cdot 120 \cdot 7,5 \cdot 0,95}{0,98} = 776 \text{ грн.},$$

W_i – встановлена потужність обладнання, кВт; t_i – тривалість роботи обладнання на етапі дослідження, год.; Це – вартість 1 кВт електроенергії, грн.;

$К_{впі}$ – коефіцієнт використання потужності; ККД – коефіцієнт корисної дії обладнання.

Інші витрати. До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$Ів = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%} = (31950 + 1021) \cdot \frac{75}{100} = 24728 \text{ грн.},$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

Накладні (загальновиробничі) витрати. До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$Внзв = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%} = (31950 + 1021) \cdot \frac{105}{100} = 34620 \text{ грн.},$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

Витрати на проведення науково-дослідної роботи. Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + К_в + В_{спец} + В_{прг} + А_{обл} + В_е +$$
$$+ I_в + В_{нзв} =$$
$$= 31950 + 1021 + 3297 + 7979 + 12815 + 26164 + 816 + 776 + 24728 + 34620 =$$
$$= 144166 \text{ грн.}$$

Загальні витрати. Загальні витрати ЗВ на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} = \frac{144166}{0{,}9} = 160184 \text{ грн.},$$

де η – коефіцієнт, що характеризує етап виконання науково-дослідної роботи. Оскільки, якщо науково-технічна розробка знаходиться на стадії впровадження технології, то η=0,9.

4.3Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

В даному випадку відбувається розробка засобу, тому основу майбутнього економічного ефекту буде формувати: ΔN – збільшення кількості споживачів, яким надається відповідна інформаційна послуга в аналізовані

періоди часу; N – кількість споживачів, яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки; $Ц_б$ – вартість послуги у році до впровадження інформаційної системи; $\pm\Delta Ц_о$ – зміна вартості послуги (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу.

Можливе збільшення чистого прибутку у потенційного інвестора ΔПі для кожного із років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховується за формулою 4.7:

$$\Delta П = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N_i)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\vartheta}{100}\right), \quad (4.7)$$

де ±ΔЦ – зміна основного якісного показника від впровадження результатів науково-технічної розробки в аналізованому році. Зазвичай, таким показником може бути зміна ціни реалізації одиниці нової розробки в аналізованому році (відносно року до впровадження цієї розробки); $\pm\Delta Ц_о$ може мати як додатне, так і від'ємне значення (від'ємне – при зниженні ціни відносно року до впровадження цієї розробки, додатне – при зростанні ціни); N – основний кількісний показник, який визначає величину попиту на аналогічні чи подібні розробки у році до впровадження результатів нової науково-технічної розробки; $Ц_о$ – основний якісний показник, який визначає ціну реалізації нової науково-технічної розробки в аналізованому році; $Ц_б$ – основний якісний показник, який визначає ціну реалізації існуючої (базової) науково-технічної розробки у році до впровадження результатів; ΔN – зміна основного кількісного показника від впровадження результатів науково-технічної розробки в аналізованому році. Зазвичай таким показником може бути зростання попиту на науково-технічну розробку в аналізованому році (відносно року до впровадження цієї розробки); λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2023 році ставка

податку на додану вартість становить 20%, а коефіцієнт λ =0,8333; ρ – коефіцієнт, який враховує рентабельність інноваційного продукту (послуги). Рекомендується брати ρ =0,2...0,5; ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році ϑ =18%.

Очікуваний термін життєвого циклу розробки 1 рік, тому:

$$\Delta П = \big((27000 - 12000) \cdot 150 - (200 - 150) \cdot 12000\big) \cdot 0{,}8333 \cdot 0{,}3 \cdot \left(1 - \frac{18}{100}\right) = 329868 \text{ грн.}$$

Далі розраховують приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t} = \frac{329868}{(1+0{,}1)^1} = 299880 \text{ грн.},$$

де $\Delta П$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн.; Т – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки (приймаємо Т=1 рік); τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,05…0,15; t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

Далі розраховують величину початкових інвестицій PV , які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{інв} \cdot ЗВ = 1 \cdot 160184 = 160184 \text{ грн.}$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{інв}$=1...5, але може бути і більшим; ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

Тоді абсолютний економічний ефект $Е_{абс}$ або чистий приведений дохід для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$Е_{абс} = ПП - PV = 299880 - 160184 = 139696 \text{ грн.},$$

де ПП – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, грн.; PV – теперішня вартість початкових інвестицій, грн.

Оскільки $Е_{абс}$>0, то можемо припустити про потенційну зацікавленість інвесторів у розробці.

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність Ев або показник внутрішньої норми дохідності вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Внутрішня економічна дохідність інвестицій Ев, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, розраховується за формулою:

$$Е_в = \sqrt[Т_ж]{1 + \frac{Еабс}{PV}} = \sqrt[1]{1 + \frac{139696}{160184}} = 1{,}36,$$

де $Т_ж$ – життєвий цикл розробки, роки.

Визначимо бар'єрну ставку дисконтування $\tau_{мін}$, тобто мінімальну внутрішню економічну дохідність інвестицій, нижче якої кошти у впровадження науково-технічної розробки та її комерціалізацію вкладатися не будуть.

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$ визначається за формулою:

$$\tau_{мін} = d + f = 0{,}9 + 0{,}05 = 0{,}95,$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d =0,9...0,12; f – показник, що характеризує ризикованість вкладення інвестицій; зазвичай величина f =0,05...0,5, але може бути і значно вищою.

Оскільки $Е_в$=1,36>$\tau_{мін}$=0,95, то потенційний інвестор може бути зацікавлений у фінансуванні впровадження науково-технічної розробки та виведенні її на ринок, тобто в її комерціалізації.

Далі розраховуємо період окупності інвестицій $Т_о$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$Т_о = \frac{1}{Е_в} = \frac{1}{1{,}36} = 0{,}74 \text{ року.}$$

Оскільки $Т_о$=0,74<1…3-х років, то це свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження цієї розробки та виведення її на ринок.

ВИСНОВКИ

У магістерській кваліфікаційній роботі розроблено інформаційну технологію контролю прийняття ліків з підвищеною точністю контролю завдяки використанню експертної системи та розширено функціональні можливості системи.

Проаналізовано та обгрунтовано актуальність розробки інформаційної технології контролю прийняття ліків. Здійснено аналіз існуючих рішень та детально проаналізовано їх недоліки та сформовано теоретичну базу для розширення функціональних можливостей системи контролю прийому ліків.

Здійснено огляд та аналіз існуючих програмних рішень, що призначені для використання у сфері контролю прийняття ліків. Виявлено, що сучасні програми не забезпечують достатню точність контролю прийняття ліків.

Розроблено структурну схему інформаційної технології контролю прийняття ліків. Схема складається з двох модулів: клієнтської частини (відповідає за взаємодію з користувачем та відображення інформації) та веб-сервера (виконує роль обробника інформації та відповідає за зберігання даних), експертної системи.

Створено діаграму діяльності роботи інформаційної технології контролю прийняття ліків. Здійснено аналіз архітектурних рівнів та наведено роль кожного рівня взаємодії, а саме приклад взаємодії серверної частини з клієнським застосунком.

Обрано та обгрунтовано систему для керування базами даних MsSql, що надає можливість швидко обробляти великі масиви даних та містить усі переваги реляційної СУБД.

Створено базу даних, зі зв’язками між таблицями. Здійснено нормалізацію бази даних до третьої форми. Перша нормальна форма надала можливість позбутися повторів кортежів у сутностях бази даних інформаційної технології та створено лише прості скалярні атрибути. Друга нормальна форма була досягнута наявністю залежності кожного не ключового

атрибуту від первинного ключа. Третя нормальна форма здійснена за рахунок відокремлення всіх не ключових полів, які відносяться до декількох кортежів, у інші таблиці.

Проаналізовано та обґрунтовано вибір мови програмування JavaScript для розробки інформаційної технології контролю прийняття ліків, а також фреймворк React Native, через його крос-платформенні можливості та бездоганну інтеграцією з Node.js, що підтверджує його придатність для розробки зручного та адаптивного застосунку.

Розроблено інформаційну технологію контролю прийняття ліків із застосуванням новітніх технологій та мікросервісної архітектури. Що в свою чергу забезпечує можливість масштабування системи через розбиття на невеликі модулі.

Виконано тестування розробленого програмного додатку, що показало правильність роботи та відповідність зазначеним технічним вимогам, доведено розширення функціональних можливостей системи контролю прийняття ліків.

Доведено економічну доцільність розробки даної технології. В результаті аналізу розрахунків, розроблений програмний продукт за ціною дешевший за аналоги. Витрати на розробку нового програмного продукту складає 160184 гривень. Період окупності складе близько 0,74 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Русначенко М. О., Арсенюк І. Р. Перспективність використання мікрофронтендів для реалізації застосунку контролю прийняття ліків.// Матеріали конференції "Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення (випуск 82): матеріали Міжнародної наукової інтернет-конференції, (м. Тернопіль, Україна, м. Ополе, Польща, 9 - 10 листопада 2023 р.) / редкол. : О. Патряк та ін. ГО “Наукова спільнота”, WSZIA w Opolu. Тернопіль : ФО-П Шпак В. Б. 2023. 206 с. С. 75 – 77.
2. Medication Dispensing Errors and Prevention. URL: https://www.ncbi.nlm.nih.gov/books/NBK519065/
3. Що таке система eHealth. URL: https://ehealth.gov.ua.
4. eHealth: що дасть українцям електронна система охорони здоров’я? URL: https://emci.ua/iak-pidkliuchytysia-do-ehealth.
5. Додаток для прийму ліків Medisafe. URL: https://www.medisafeapp.com.
6. Додаток для прийму ліків Dosecast. URL: http://www.montunosoftware.com/about.
7. Додаток для прийму ліків CareClinic. URL: https://careclinic.io.
8. Додаток для прийму ліків Lady Pill Reminder. URL: https://baviux.com/app/lady-pill-reminder.
9. Microsoft SQL Server URL: https://www.microsoft.com/en-us/sql-server.
10. Transact-SQL URL: https://en.wikipedia.org/wiki/Transact-SQL.
11. Expert system URL: https://en.wikipedia.org/wiki/Expert_system.
12. Java. URL: https://uk.wikipedia.org/wiki/Java.
13. C Sharp. URL: https://uk.wikipedia.org/wiki/C_Sharp.
14. JavaScript. URL: https://developer.mozilla.org/ru/docs/Web/JavaScript.
15. React Native. URL: https://reactnative.dev.
16. NativeScript. URL: https://nativescript.

17. Ionic. URL: https://ionic.io.
18. Monolithic Architecture. URL: https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith.
19. Microservice Architecture. URL: https://cloud.google.com/learn/what-is-microservices-architecture.
20. ModuleFederationPlugin. URL: https://webpack.js.org/plugins/module-federation-plugin.
21. DOM. URL: https://developer.mozilla.org/docs/Web/API/Document-Object-Model.
22. CORS. URL: https://developer.mozilla.org/ru/docs/Web/HTTP/CORS.
23. Node.js. URL: https://nodejs.org/en.
24. Express.js. URL: https://expressjs.com.
25. MongoDB. URL: https://www.mongodb.com/docs.
26. Солоний М., Арсенюк І. Розоробка мобільного додатку інтернет-магазину за допомогою React Native // Матеріали XLVIII науково-технічної конференції факультету інформаційних технологій та комп'ютерної інженерії ВНТУ, Вінниця, 2019. URL: https://conferences.vntu.edu.ua/index.php/allfitki/all-fitki-2019/paper/view/7012/5949
27. Push notifications React Native. URL: https://blog.logrocket.com/how-to-create-and-send-push-notifications-in-react-native.
28. 10 найкращих додатків для прийому ліків. URL: https://www.medicalnewstoday.com/articles/pill-reminder.
29. 6 of the Best Reminders for Your Medications. URL: https://www.healthline.com/health/best-medication-reminders.
30. 7 найкращих практик для застосунків React Native. URL: https://javascript.plainenglish.io/7-best-practices-for-react-native-applications-be1dd907e657.

Додаток А (обов'язковий)

Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Інформаційна технологія контролю прийняття ліків

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 89,1% Схожість 10,9%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи Русначенко М.О.

Керівник роботи Арсенюк І.Р.

Опис прийнятого рішення

Магістерську кваліфікаційну роботу допущено до захисту

Особа, відповідальна за перевірку Озеранський В.С.

Додаток Б (обов’язковий)

Лістинг програми

Вхідна точка проекту

```
import React from "react";
import {StatusBar} from 'expo-status-bar';
import {NavigationContainer} from '@react-navigation/native';
import {AuthContextProvider} from "./src/contexts/AuthContext";
import {Router} from "./src/navigation/Router";

export default function App() {
   return (
      <NavigationContainer>
         <AuthContextProvider>
            <StatusBar barStyle="default" backgroundColor="#359376"/>
            <Router />
         </AuthContextProvider>
      </NavigationContainer>
   );
}
```

Модуль сповіщень

```
import React, {useEffect, useState, useRef} from "react";
import {Modal, StyleSheet, TouchableHighlight} from "react-native";
import Fontisto from "react-native-vector-icons/Fontisto";
import {CreateTreatmentForm} from "./CreateTreatmentForm";
import * as Notifications from 'expo-notifications';
import * as Device from 'expo-device';

Notifications.setNotificationHandler({
```

```
    handleNotification: async () => ({
        shouldShowAlert: true,
        shouldPlaySound: false,
        shouldSetBadge: false,
    }),
});

export const CreateTreatment = () => {
    const [modalVisible, setModalVisible] = useState(false);
    const [expoPushToken, setExpoPushToken] = useState('');
    const [notification, setNotification] = useState(false);
    const notificationListener = useRef();
    const responseListener = useRef();

    const closeModal = () => {
        setModalVisible(false);
    }

    useEffect(() => {
        registerForPushNotificationsAsync().then(token =>
setExpoPushToken(token));

        // This listener is fired whenever a notification is received while the app is
foregrounded
        notificationListener.current =
Notifications.addNotificationReceivedListener(notification => {
            setNotification(notification);
        });
```

```
      // This listener is fired whenever a user taps on or interacts with a notification
(works when app is foregrounded, backgrounded, or killed)
      responseListener.current =
Notifications.addNotificationResponseReceivedListener(response => {
        console.log(response);
      });

      return () => {
        Notifications.removeNotificationSubscription(notificationListener.current);
        Notifications.removeNotificationSubscription(responseListener.current);
      };
    }, []);

    return (
      <>
        <TouchableHighlight
          style={styles.button}
          onPress={() => setModalVisible(true)}
        >
          <Fontisto name="plus-a" size={40} color={"#fff"} />
        </TouchableHighlight>

        <Modal
          animationType="slide"
          transparent={false}
          visible={modalVisible}
          onRequestClose={closeModal}
        >
          <CreateTreatmentForm closeModal={closeModal} />
        </Modal>
```

```
      </>
    )
}

async function sendPushNotification(expoPushToken) {
    const message = {
      to: expoPushToken,
      sound: 'default',
      title: 'Original Title',
      body: 'And here is the body!',
      data: { someData: 'goes here' },
    };

    await fetch('https://exp.host/--/api/v2/push/send', {
      method: 'POST',
      headers: {
        Accept: 'application/json',
        'Accept-encoding': 'gzip, deflate',
        'Content-Type': 'application/json',
      },
      body: JSON.stringify(message),
    });
}

async function registerForPushNotificationsAsync() {
    let token;
    if (Device.isDevice) {
      const { status: existingStatus } = await Notifications.getPermissionsAsync();
      let finalStatus = existingStatus;
      if (existingStatus !== 'granted') {
```

```
            const { status } = await Notifications.requestPermissionsAsync();
            finalStatus = status;
        }
        if (finalStatus !== 'granted') {
            alert('Failed to get push token for push notification!');
            return;
        }
        token = (await Notifications.getExpoPushTokenAsync()).data;
        console.log(token);
    } else {
        alert('Must use physical device for Push Notifications');
    }

    if (Platform.OS === 'android') {
        Notifications.setNotificationChannelAsync('default', {
            name: 'default',
            importance: Notifications.AndroidImportance.MAX,
            vibrationPattern: [0, 250, 250, 250],
            lightColor: '#FF231F7C',
        });
    }

    return token;
}
```

Провайдер авторизації

```
import {JwtPayload} from "../definitions/types";
import React, {createContext, useContext, useEffect, useState} from "react";
import {validateTokenRequest} from "../api/auth.api";
```

```
import * as cookies from 'js-cookie';
import {useHistory} from "react-router";
import {decodeJwtToken} from "../helpers/authHelpers";
import {UserRole} from "../definitions/user-roles";

const AuthContext = createContext(null);

export const AuthContextProvider = ({ children }) => {
  const [roles, setRoles] = useState([]);
  const [userId, setUserId] = useState(0);
  const [validated, setValidated] = useState(false);

  const history = useHistory();

  const handleHomeNavigation = () => {
    history.push('/');
  }

  const handleAuthEndpointNavigation = () => {
    return window.location.href =
`${process.env.REACT_APP_API_URL}api/auth/${
        encodeURIComponent(`${window.location.origin}/auth`)
    }`;
  }

  const handleQueryAuthorization = async () => {
    const query = new URLSearchParams(window.location.search);
    const token = query.get('token');

    if (token) {
```

```
    if (await validateTokenRequest(token)) {
      cookies.set('token', token, { expires: 365 });
      handleHomeNavigation();
    } else {
      handleAuthEndpointNavigation();
    }

    return true;
  }

  return false;
}

const handleCookiesAuthorization = async () => {
  const token = cookies.get('token');

  if (!token) {
    handleAuthEndpointNavigation();
    return;
  }

  const validationResult = await validateTokenRequest(token);

  if (!validationResult) {
    handleAuthEndpointNavigation();
  }
}

const handleMapCookiesToState = () => {
  const token = cookies.get('token');
```

```
  if (token) {
    const jwtData = decodeJwtToken<JwtPayload>(token);

    setRoles(
        Array.isArray(jwtData.role) ? jwtData.role : [jwtData.role]
    );

    setUserId(+jwtData.nameid)
  }
}

useEffect(() => {
  (async () => {
    const queryAuthorizationResult = await handleQueryAuthorization();

    if (!queryAuthorizationResult) {
      await handleCookiesAuthorization();
    }

    handleMapCookiesToState();

    setValidated(true);
  })()
}, []);

const hasAnyRole = (requiredRoles) => {
  return roles.some(role=>requiredRoles.includes(role));
}
```

```
  if (!validated) {
    return <></>;
  }

  return (
    <AuthContext.Provider value={{ roles, userId, hasAnyRole }}>
      { children }
    </AuthContext.Provider>
  )
}

export const useAuthContext = () => {
  const state = useContext(AuthContext)

  if (!state) {
    throw new Error('Hook cannot be used outside AuthContext Provider');
  }

  return state;
};

export const getAuthTokenRequest = async (): Promise<AuthResponseDto> => {
  return await fetch(`${process.env.REACT_APP_API_URL}api/auth`, {
    credentials: 'include'
  }).then((response) => response.json());
}

export const validateTokenRequest = async (token: string): Promise<boolean> =>
```

```
fetch(`${process.env.REACT_APP_API_URL}api/auth/validate?token=${token}`
)
   .then(response => response.json())
   .catch(error=> {
     if(error){
       console.error(error);
     }
     return false;
   });
```

Модуль авторизації

```
import {StyleSheet, Text, TextInput, TouchableHighlight, View} from "react-
native";
import React, {useState} from "react";
import {useAuthContext} from "../../contexts/AuthContext";

export const SignUp = () => {
   const [login, setLogin] = useState("");
   const [password, setPassword] = useState("");
   const [confirmedPassword, setConfirmedPassword] = useState("");
   const [error, setError] = useState(false);
   const { setIsAuth } = useAuthContext();

   const signUp = () => {
      if (login && password === confirmedPassword) {
         setIsAuth(true);
         setError(false);
      } else {
```

```
            setError(true);
        }
    }

    return (
        <View style={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>
            <Text style={{fontSize: 24, color: "#359376", fontWeight:
"bold"}}>Зареєструватись</Text>
            <TextInput
                style={[styles.input, error && styles.errorInput]}
                onChangeText={setLogin}
                value={login}
                placeholder={"Логін"}
            />
            <TextInput
                style={[styles.input, error && styles.errorInput]}
                onChangeText={setPassword}
                value={password}
                placeholder={"Пароль"}
                secureTextEntry={true}
            />
            <TextInput
                style={[styles.input, error && styles.errorInput]}
                onChangeText={setConfirmedPassword}
                value={confirmedPassword}
                placeholder={"Підтвердіть пароль"}
                secureTextEntry={true}
            />
            <TouchableHighlight
                style={styles.button}
```

```
            onPress={signUp}
          >
            <Text style={{color: "#fff", fontSize: 16}}>Зареєструватись</Text>
          </TouchableHighlight>
        </View>
    )
}

import React, {useState} from "react";
import {Text, View, TextInput, StyleSheet, TouchableHighlight} from "react-
native";
import {useAuthContext} from "../../contexts/AuthContext";

export const SignIn = ({ navigation }) => {
    const [login, setLogin] = useState("");
    const [password, setPassword] = useState("");
    const [error, setError] = useState(false);
    const { setIsAuth } = useAuthContext();

    const signIn = () => {
        if (login === "mru@gmail.com" && password === "123456") {
            setIsAuth(true);
            setError(false);
        } else {
            setError(true);
        }
    }

    return (
        <View style={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>
```

```
        <Text style={{fontSize: 24, color: "#359376", fontWeight:
"bold"}}>Увійти</Text>
        <TextInput
          style={[styles.input, error && styles.errorInput]}
          onChangeText={setLogin}
          value={login}
          placeholder={"Логін"}
        />
        <TextInput
          style={[styles.input, error && styles.errorInput]}
          onChangeText={setPassword}
          value={password}
          placeholder={"Пароль"}
          secureTextEntry={true}
        />
        <TouchableHighlight
          style={styles.button}
          onPress={signIn}
        >
          <Text style={{color: "#fff", fontSize: 16}}>Увійти</Text>
        </TouchableHighlight>
        <TouchableHighlight
          style={[styles.button, styles.createButton]}
          onPress={() => navigation.navigate("Зареєструватись")}
        >
          <Text style={{color: "#359376", fontSize: 16}}>Створити
акаунт</Text>
        </TouchableHighlight>
      </View>
    )
```

Додаток В (обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ КОНТРОЛЮ ПРИЙНЯТТЯ ЛІКІВ

Виконав: студент 2-го курсу,
групи 1КН-22м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
_____ Русначенко М.О.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН
_____ Арсенюк І.Р.
(прізвище та ініціали)
«_07_»_____12_____2023 р.

Вінниця ВНТУ - 2023 рік

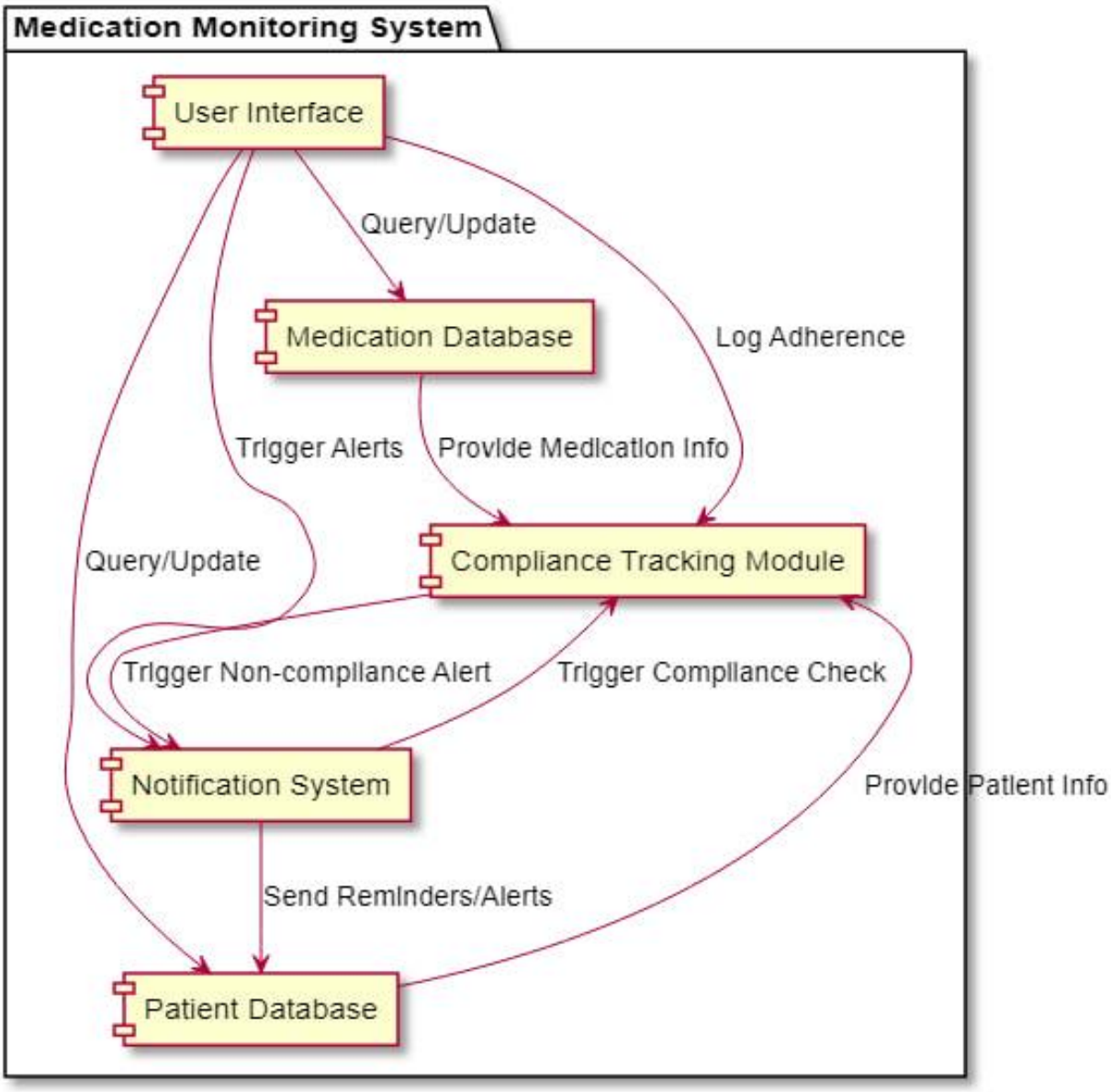

Рисунок В.1 – Структурна схема системи контролю прийому ліків

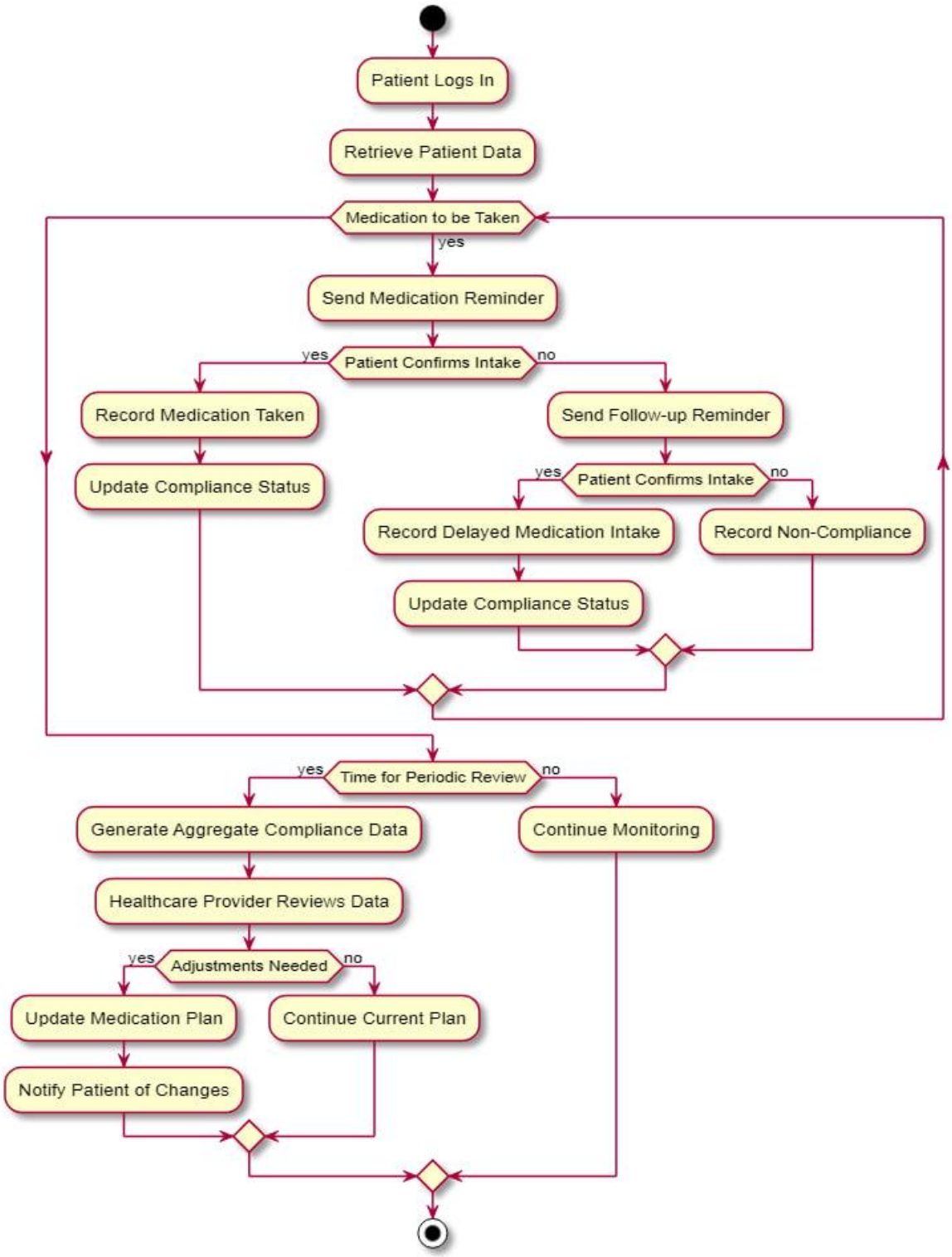

Рисунок В.2 – Діаграма діяльності роботи програмного забезпечення

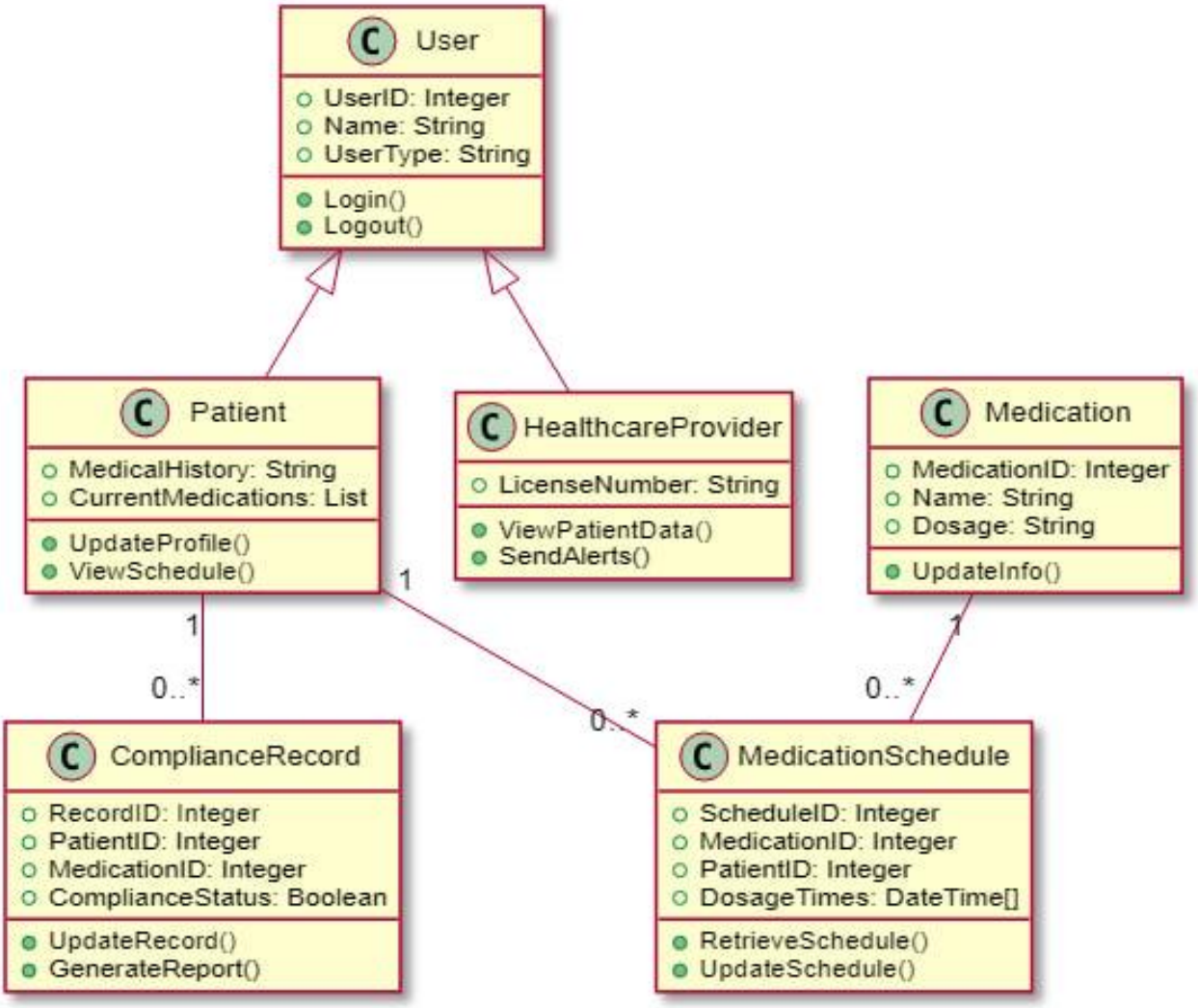

Рисунок В.3 – UML-діаграма класів

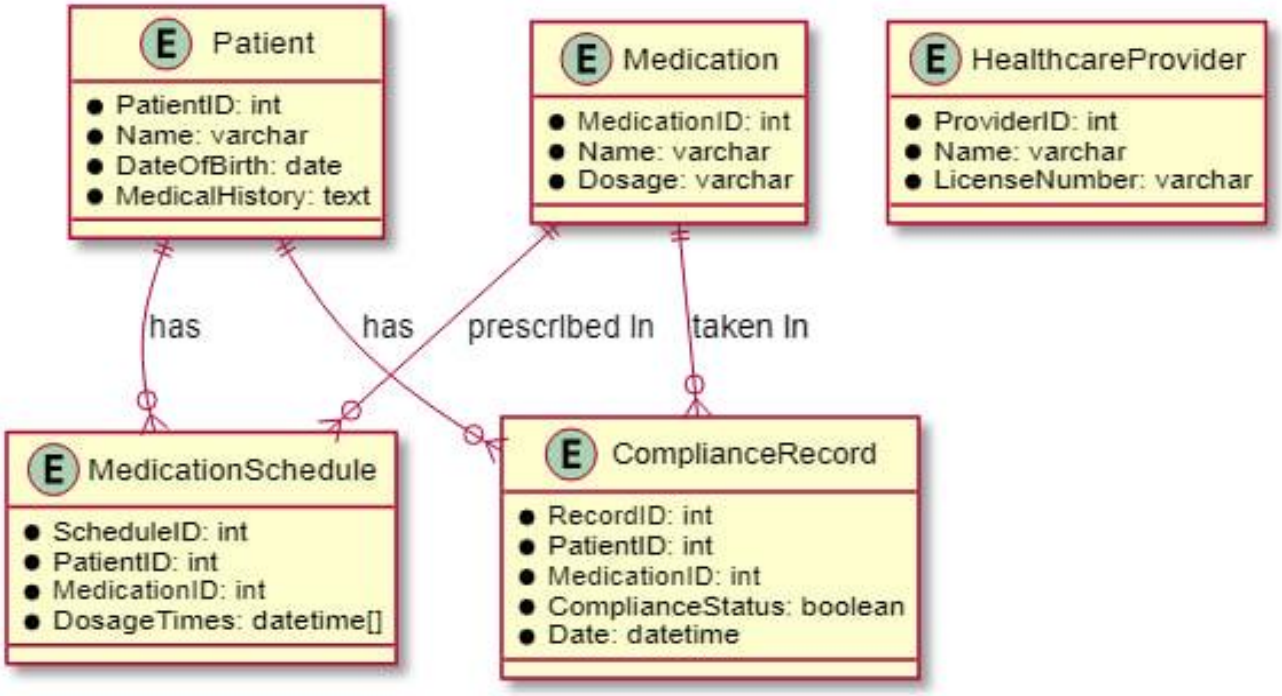

Рисунок В.4 – Загальна схема бази даних роботи системи

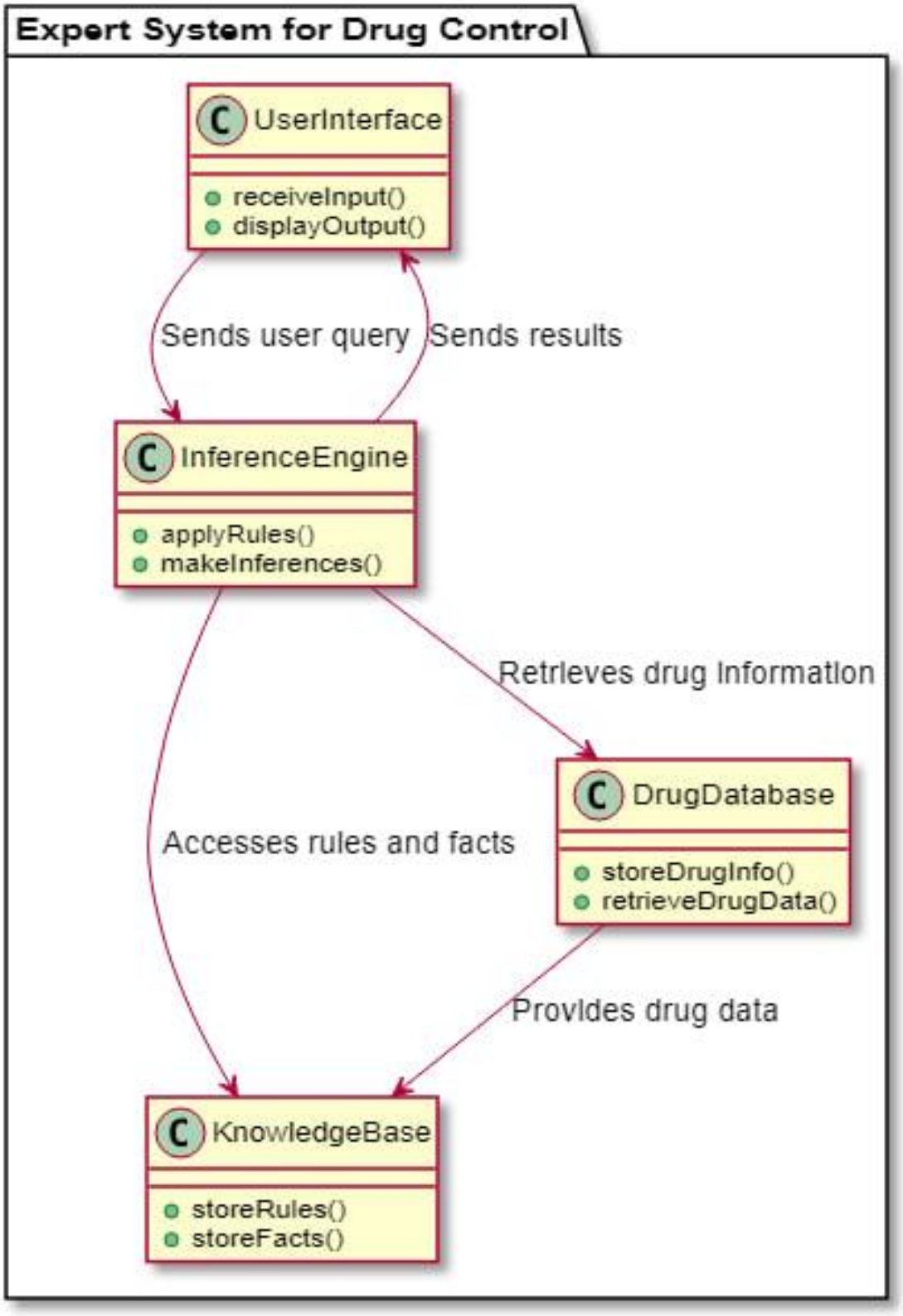

Рисунок В.5 – Загальна схема роботи експертної системи

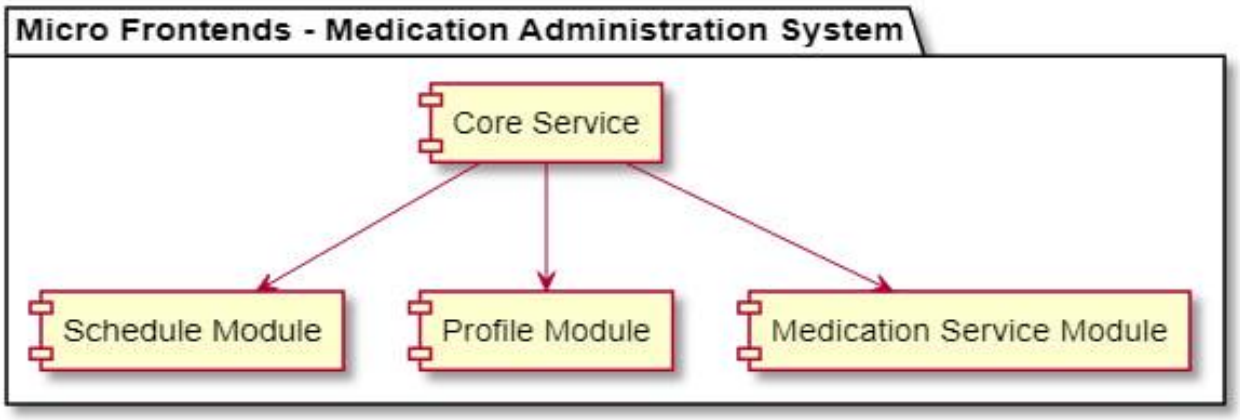

Рисунок В.6 – Схема мікросервісної архітектури клієнтської частини застосунку